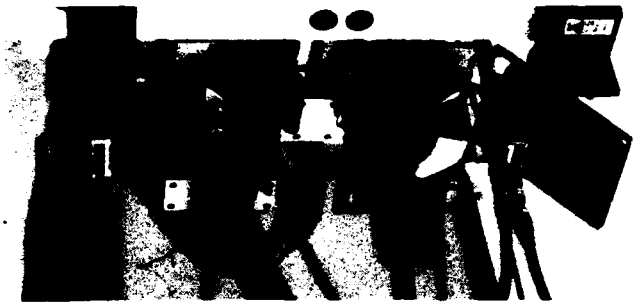
Volume I

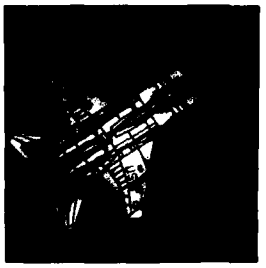
PROCEEDINGS:

# Image Understanding Workshop












**Sponsored by:**

Defense Advanced Research Projects Agency
Information Science and Technology Office

DARPA

SC

**April 1988**

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

| | |
|---|---|
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle)<br>Proceedings:<br>Image Understanding Workshop<br>April 1988 | 5. TYPE OF REPORT & PERIOD COVERED<br>ANNUAL TECHNICAL<br>February 1987-April 1988<br>6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s)<br>Lee S. Baumann (Editor) | 8. CONTRACT OR GRANT NUMBER(s)<br>N00014-86-C0700 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>SCIENCE APPLICATIONS INTERNATIONAL CORPORATION<br>1710 Goodridge Drive<br>McLean, VA 22102 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS<br>ARPA Order 5605 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Defense Advanced Research Projects Agency<br>1400 Wilson Blvd<br>Arlington, VA 22209 | 12. REPORT DATE<br>April 1988<br>13. NUMBER OF PAGES<br>1165 (2 Vols.) |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) | 15. SECURITY CLASS. (of this report)<br>UNCLASSIFIED<br>15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELASE, DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital Image Processing; Image Understanding; Scene Analysis; Edge Detection; Image Segmentation; CCDArrays; CCD Processors

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document contains the annual progress reports and technical papers presented by the research activities in the Image Understanding, sponsored by the Information Science & Technology Office, Defense Advanced Research Projects Agency. The papers were presented at a workshop conducted on 6-8 April 1988, in Cambridge, Massachusetts. Also included are copies of invited papers presented at the workshop and additional technical papers from the research activities which were not presented due to lack of time but are germane to this research field.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

# Image Understanding Workshop

Proceedings of a Workshop
Held at
Cambridge, Massachusetts

**April 6-8, 1988**

Volume I





Sponsored by:

***Defense Advanced Research Projects Agency***
**Information Science and Technology Office**

**This document contains copies of reports prepared for the DARPA Image Understanding Workshop. Included are results from both the basic and strategic computing programs within DARPA/ISTO sponsored projects.**

Distributed by

Morgan Kaufmann Publishers, Inc.

2929 Campus Drive

San Mateo, California 94403

ISBN 0-934613-68-0

Printed in the United States of America

TABLE OF CONTENTS

**TABLE OF CONTENTS**

TABLE OF CONTENTS

TABLE OF CONTENTS

## TABLE OF CONTENTS

**TABLE OF CONTENTS**

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

**TABLE OF CONTENTS**

## AUTHOR INDEX

**AUTHOR INDEX** (Continued)

## AUTHOR INDEX (Continued)

## AUTHOR INDEX (Continued)

## Acknowledgments

This workshop was organized at the direction of Ltc. Robert L. Simpson, Program Manager for Machine Intelligence in the Information Science and Technology Office of the Defense Advanced Research Projects Agency (DARPA). The main theme of the 1988 workshop, the 18th in this DARPA sponsored series of meetings on Image Understanding and Computer Vision, is "Parallel Architectures and Algorithms." Sessions are planned to cover new vision techniques in prototype vision systems for manufacturing, navigation, cartography, and photointerpretation. As usual the main objectives of the workshop are to review the latest research results in the DARPA IU research progam, help keep the government research community aware of evolving technology and most importantly to exchange ideas, needs and trends in computer vision research. Dr. Tomaso Poggio of the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology coordinated the technical program for this workshop.

As part of the agenda principal investigator's reports and technical papers are scheduled to be presented from MIT, University of Southern California, Stanford University, University of Maryland, Carnegie-Mellon University, SRI International, University of Massachusetts, Rochester University, Columbia University, Hughes Research Laboratory, General Electric Research and Development Center, Advanced Decision Systems, and Honeywell Systems and Research Center. In addition, invited papers are planned for presentation from the University of Illinois, NYU - Courant Institute, University of Washington, University of Pennsylvania, David Sarnoff Research Center, and Schlumber Palo Alto Research Center.

As is the normal practice for these workshops all technical reports submitted by participating organizations are published in this comprehensive proceedings available for distribution to the attendees, including both those presented at the workshop and those for which lack of time precluded presentation.

The figure appearing on the cover was prepared by Dr. J. Little of the MIT AI Laboratory and represents a block diagram of the integration stage of the MIT Vision Machine. For details see "The Vision Machine", authored by Dr. T. Poggio, et. al., in these proceedings. The cover layout was prepared by Mr. Tom Dickerson of the Science Applications International Corporation (SAIC) graphic arts staff. Appreciation is also extended to Ms. Bethany Moss of SAIC for her work in compiling the papers into this proceedings and handling the mailings associated with the 1988 I. U. Workshop.

# SECTION I

# PROGRAM REVIEWS BY PRINCIPAL INVESTIGATORS

# MIT PROGRESS IN UNDERSTANDING IMAGES

T. Poggio and the staff

The Artificial Intelligence Laboratory
Massachusetts Institute of Technology

**ABSTRACT**

*Our work in the past year has concentrated on three main projects, each one representing a complementary aspect of a complete vision system. The first project – a parallel Vision Machine – has the goal of developing a system for integrating early vision modules and computing a robust description of the discontinuities of the surfaces and of their physical properties. Additional goals of the project are the refinement of early vision algorithms and their implementation on a massively parallel architecture such as the Connection Machine System. The second project concerns visual recognition: we have developed several schemes for model based recognition and implemented them. Finally, we have continued our work in autonomous navigation. Around these main themes, additional work, at the theoretical and implementation level, has been done in motion analysis, navigation, photogrammetry, visual routines and learning.*

## 1. Introduction

This report reviews the main results of our work in Image Understanding during the past year. We will first outline our main projects in vision and then sketch a few of the other smaller projects. The first project is focused on the problem of integrating different early vision algorithms to produce a cartoon-like description of the discontinuities in the surfaces and their physical properties. The second major effort is in the areas of model-based recognition. It involves the use of both 2D and 3D data and models. These two projects are related. We plan to use the output of the integration stage as input to the recognition algorithms. The third project we will describe has a different ultimate goal: autonomous navigation. Finally, we will discuss some of the other studies that are somewhat less directly related to the three main projects.

## 2. The Vision Machine and Parallel Integration

The Vision Machine is a computer system that attempts to integrate several visual cues to achieve high performance in unstructured environments for the tasks of recognition and navigation. It is also a test-bed for our progress in the theory of early vision algorithms, their parallel implementation, and their integration. The Vision Machine consists of a movable two-camera Eye-Head system – the input device – and a 16K Connection Machine – our main computational engine. We have developed and implemented several parallel early vision algorithms computing edge detection, stereo, motion, texture and color in close to real time. The integration stage is based on coupled Markov Random Field models, and attempts to derive a map of the surface discontinuities in the scene, with a partial labeling of the intensity edges in terms of their physical origin. Thus the project has several complementary goals: it attempts to develop a theory of visual integration and to test it in an unstructured environment; it aims to refine and implement robust early vision algorithms in a massively parallel architecture; and it tries to build a full vision system. A rather detailed description of the present state of the project and its initial promising results is given in another paper in these Proceedings.

## 3. Object Recognition

In earlier reports, we have described several approaches to the problem of object recognition. Our work has proceeded along a number of fronts.

### 3.1. Recognition from Matched Dimensionalities

Earlier reports described the work of Grimson and Lozano-Pérez on the recognition of occluded objects from noisy sensory data under the condition of matched dimensionality. Specifically, if the objects to be recog-

nized and localized are laminar and lie on a flat surface, or if the objects are volumetric but lie in stable configurations on a flat surface, then the sensory data need only be two-dimensional (e.g. a single image); if the objects to be recognized and localized are volumetric and lie in arbitrary positions, then the sensory data must be three dimensional (e.g. stereo or motion data, laser range data). The original technique (called RAF) was designed to recognize polyhedral objects from simple measurements of the position and surface orientation of small patches of surface. The technique searches for consistent matchings between the faces of the object models and the sensory measurements, using constraints on the relative shape of pairs of model faces and pairs of measurements to reduce the search.

In the past year we have considered a number of problems in recognition associated with this approach. First, we have completed several extensions of the system to deal with different classes of objects. We have extended the two dimensional system to recognize objects composed of circular or straight boundary segments. To extract such segments from edge descriptions of the image, we transform the edge pixels into an arclength-orientation space, and then use standard split-and-merge techniques to extract the straight segments in this transformed representation. From these segments, we can derive information about the position and orientation of straight segments in the original edge description, and about the center, radius and angular extent of circular arcs in the original edge description. Recognition uses an extended version of the constrained search method of RAF, with simple pairwise constraints about the relative shape of circular segments as well as linear ones. In three dimensions, we have considered extensions to deal with simple curved surfaces, in particular, objects that can be locally approximated by cylinders and cones. In this case, we process three dimensional sensory data, such as can be obtained from laser striping systems, to deduce rulings on the surface of an object. From the rulings, the characteristics of the axes of the cylinders or cones can be derived. These are then matched, using the RAF formalism, to identify the pose of these 3D objects.

We have also completed an extension of the system to deal with some classes of parameterized objects. The first set of extensions includes the recognition of objects that can scale in size, the recognition of objects that are composed of rigid subparts connected through rotational degrees of freedom (e.g. a pair of scissors) and the recognition of objects that can undergo a stretching deformation along one axis. In each case, one can derive expressions for the geometric relationship between two edges as a function of the free parameters of the class of objects. For these kinds of parameterization, the RAF system can be extended by modifying the tree search process to pass along the range of feasible values for the free parameters as the tree of interpretations is explored. In an alternate approach, Gil Ettinger has developed and tested a system for recognizing objects that are composed of rigid subparts that can scale, rotate and translate relative to one another in composing an object. The system uses a hierarchical representation of objects, based on the Curvature Primal Sketch of Brady and coworkers [Asada and Brady, 1984]. The matching process between a model and processed sensory data uses a variation of the constraints of the RAF system. By considering objects as being composed of subparts, Ettinger has developed a method that allows for efficient and correct indexing into an automatically generated model library, so that his system can deal with a variety of objects at one time. In addition, the use of scale and object hierarchies allows the system to deal easily with objects that scale in size.

In a different vein, we have considered some theoretical implications of the constrained search approach to recognition. Using a combinatorial analysis, we have established some bounds on the expected performance of an RAF style of system, both in terms of the number of interpretations, and in terms of the amount of search required. We have shown that in the case of data known to come from a single object, the expected number of interpretations (barring symmetries of the object) asymptotically approaches 1 as the number of sensory data points is increased, where the convergence occurs for as few as three data points. We have also shown that in this case, the expected amount of search involves explicitly considering $ms$ nodes of the tree, where $m$ is the number of faces in the object model and $s$ is the number of sensory data points. When we allow for data from multiple objects, so that some of the data points are spurious, the results are less strong. We have shown that the expected number of interpretations is bounded by

$$2^c + ms + [1 + k]^s$$

where $k$ is a constant that depends on the size of the object and on the amount of error in the sensory measurements, and where $c$ is the number of sensory data points that are actually on the object of interest, out of the $s$ total data points. Since any subset of a feasible interpretation is also a feasible interpretation, the $2^c$ term represents the power set of the correct interpretation, and this bound implies that in general the only interpretation of length $c$ will be the correct one. The amount of search generally required to find this interpretation is given roughly by (a more precise but more complicated expression actually holds)

$$m2^c + m^s\binom{s}{2},$$

which implies that an exponential amount of search is needed, although the $2^c$ term is considerably reduced from the general case of $(m+1)^s$, which holds for unconstrained search. As part of this theoretical analysis, we have also considered the effect of using a Hough transform to presort subspaces of the search space to explore. We have shown that the Hough transform reduces the search needed by reducing the parameters $m$ and $s$ in the above expression. We have also shown that noise characteristics of the Hough transform, in the presence of noisy data and non-infinitesimal hash buckets, can lead to a significant probability that the Hough buckets with the largest scores may not correspond to the correct interpretation, and that the expected number of spurious pairs hashed into the correct Hough bucket is non-trivial. This implies that one should not, in general, rely on the Hough transform to fully solve the recognition problem, but rather that one should use it as a preprocessor, selecting out small subspaces within which the RAF method can be applied effectively.

Much of our earlier work with the RAF recognition system dealt with robotics environments and the recognition of industrial parts. Recently, we have begun a pilot study of applying the technique to a very different domain, underwater localization. Specifically, we have considered the problem of determining the location of an autonomous underwater vehicle by matching sensory data obtained by the vehicle against bathymetric or other maps of the environment. Sensor modalities include active methods such as sonar, and passive methods such as pressure readings and doppler data from passing ships. We have conducted some early simulation experiments using RAF, together with strategies for acquiring sensory data to solve this localization problem, with excellent results.

One of the difficulties with the RAF approach to recognition is that it does not deal with the issue of segmentation of the data in a reasonable way. In part, this is reflected in the theoretical analysis, in which the amount of search increases dramatically when spurious data is allowed. While the Hough transform can help reduce this problem, it is model driven, and hence potentially very expensive when applied to large libraries of objects. As an alternative to this, David Jacobs has directly addressed the issue of grouping in an image. Jacobs has derived measures for determining the probability that a set of edge fragments in an image is likely to have come from a single object. These measures consider simple measurements such as the separation of groups of edges, and the relative alignment of groups of edges. The recognition system, since it does not directly consider the object model, may occasionally be incorrect. However, tests of the system on a variety of images of two-dimensional and three-dimensional scenes shows a remarkable and dramatic reduction in the search required to recognize objects from a library, and also is quite effective at identifying groups of edges coming from a single object. The effect of this grouping mechanism is particularly apparent when applied to libraries of objects, since the parameters computed by the grouping scheme can be used to do effective indexing into a library.

A separate issue for recognition algorithms concerns the possibility of using parallel architectures, such as the Connection Machine, to obtain significant performance improvements. Todd Cass has completed the development and implementation of a parallel recognition scheme for two dimensional scenes. The system uses a careful Hough transform method, followed by a sampling scheme in the parameter space to find instances of an object and its pose. Typical performance of the method involves the correct identification and localization of heavily occluded objects, in scenes in which a large number of other parts are present, in under five seconds, using a 16K processor configuration of the Connection Machine.

### 3.2. Recognition Under Projection

All of the previous work has been restricted to the domain of matched dimensionality. Another problem concerns the recognition of solid objects undergoing six degrees of positional freedom from a single two-dimensional image. In the last Proceedings, we reported on an approach by Dan Huttenlocher and Shimon Ullman for addressing this problem.

Huttenlocher and Ullman have shown that a correspondence between three points on a rigid solid object and three points in a two-dimensional image is sufficient to *align* the object with the image. The method assumes a "weak perspective" viewing model, where true perspective is approximated by orthographic projection plus a scale factor. The alignment transformation specifies the three-dimensional rotation, the two-dimensional translation, and the scale factor that bring an object model into correspondence with an image. Huttenlocher and Ullman have proved that this transformation exists, and is unique up to a reflection, for any noncolinear triple of corresponding model and image points. A closed form solution is given for computing the alignment from a triple of points, two oriented points, or

three edge fragments.

To demonstrate the use of alignment in recognition, Huttenlocher and Ullman have implemented a system for recognizing solid objects with arbitrary three-dimensional position and orientation from a single two-dimensional view. The recognizer solves for potential alignments of a model and an image using features that define either two or three points. Every three-point feature and each pair of two-point features specify a possible alignment. Each of these potential positions and orientations is verified by projecting the model into the image, and counting the number of model features that lie near similar image features. This recognition algorithm has a worst case running time of $O(m^3i^2)$ for $m$ model features and $i$ image features, since every pair of model and image features may specify an alignment, and there are $m$ features to check for each alignment. The method is discussed in more detail elsewhere in these Proceedings.

## 4. A Mobile Robot

### 4.1. Integrating Motion and Stereo for Navigation

Rodney Brooks, Anita Flynn and Thomas Marill have been looking into the problem of building self calibrating vision systems for autonomous vehicle navigation. Field conditions for autonomous vehicles may be very dynamic, involving rough terrain and powerful blast events occurring intermittently. A vision system for such a system would either have to be massive and extremely strong structurally to avoid misalignment, or it would have to be rapidly self calibrating. The experiments Brooks et. al. have done have demonstrated a system that is capable of self recalibration in a few tens of frame times [Brooks, Flynn and Marill, these Proceedings].

The idea is to use one self calibrating vision process (forward motion vision) to calibrate another (stereo vision) without resorting to any external units of measurement. Both are calibrated to a velocity dependent coordinate system which is natural to the task of obstacle avoidance. The resulting vision system is continually self calibrating, making it tolerant of normal mechanical drift. But better than that, it is also tolerant of severe and sudden misalignments. After a few seconds it adapts to its grossly altered sensor alignments.

The algorithms require no pre-knowledge of camera focal lengths, fine orientation, or stereo baseline separation. With such quick calibration and adaptation the sensors can be mounted on cheap steerable systems. We can trade cheap computation for deficiencies which arise from avoiding expensive mechanical solutions to sensor steering problems.

The foundations of these algorithms, in a world of perfect measurement, are quite elementary. The contribution of this work is to make them noise tolerant while remaining simple computationally. Both the algorithms and the calibration procedure are easy to implement and have shallow computational depth, making them (1) run at reasonable speed on moderate uni-processors, (2) appear practical to run continuously, maintaining an up-to-the-second calibration on a mobile robot, and (3) appear to be good candidates for massively parallel implementations.

So far the experiments performed have used images sequences collected from a mobile platform at 7.5 stereo pairs per second followed by offline analysis. The next task is to integrate processing on board the robot. The design goal is onboard processing at a rate of 10 stereo pairs per second. Next we will take the output of the vision system and use it as input to the navigation algorithms previously demonstrated on our mobile platforms [Brooks 86].

## 5. Topics in Early and High-Level Vision

### 5.1. Direct Motion Vision

Berthold Horn is continuing to study the recovery of rigid body motion and surface shape directly from first derivatives of image brightness (these methods appear to be of great importance in "short-range" motion, while feature-based methods are more appropriate in "long-range" motion).

Several special cases have been solved, and progress has been made in suppressing problems discovered in derivative estimation caused by under-sampling in both image space and time. A very robust method has been developed for the case of pure rotation, and conclusions have been reached about which of several methods for the case of pure translation works best (this is a method that minimizes the integral of estimated depth squared over the image region). Sensitivity analysis shows the need for a wide field of view and the futility of attempts to recover full three dimensional motion from small patches. This implies, for example, that methods based on second partial derivatives of optical flow, while formally correct, lead to ill-posed problems, and are thus not useful in the presence of noise.

With John Harris at Hughes' Artificial Intelligence Center, we are working on an application of this di-

rect approach to range image sequences. In this case, a clean least-squares solution is possible without ambiguities. The method leads to a set of linear equations in the six parameters of motion. The coefficients of these equations are integrals over the image region of products of first derivatives of range with respect to image coordinates and time (range rate). The new method will be tested on data generated by the ERIM scanner.

### 5.2. Combining Long-Range and Short-Range Motion Measurements

In order to design a flexible and robust motion measurement system, it may be necessary to integrate the fast computation of image velocities with a longer range tracking of localizable image features. The short-range velocity-based system would serve to detect sudden movements, locate object boundaries defined by motion discontinuities, and provide for the rough estimate of the 3-D layout of the scene. This short-range system could also facilitate camera tracking by providing a rough indication of the direction and speed of movement of image features. The long-range tracking system would provide a more accurate measurement of the motion of image features over a longer time period, for the purpose of recovering the detailed 3-D shape of objects.

Michael Drumheller and Ellen Hildreth are currently exploring one method by which short-range motion measurements can influence the motion correspondence process. In Ullman's minimal mapping scheme for motion correspondence [Ullman, 1979], the probability that a given feature at one moment corresponds to a particular feature at a later time depends in part on the distance traveled between frames. Ullman also assumed that the probability of particular 2-D displacements (velocities) would decrease monotonically with size (speed), and would be uniformly distributed with respect to direction. The short-range velocity measurements can be used to modify these probability distributions in such a way that greater weight is given to velocities within a neighborhood of the estimated direction and speed of movement. We have implemented Ullman's minimal mapping scheme and have begun testing of the algorithm with natural images of rotating objects, with and without making use of short-range velocity information. Computer simulations show that the combination of velocity measures with motion correspondence leads to better performance at motion measurement than either strategy on its own.

### 5.3. Using Time-to-Collision Estimates

Some visual tasks, such as high-performance navigation (negotiating through narrow channels, landing aircraft, walking a tightrope) require an accurate model of the 3-D structure and motion of object surfaces, while others may require only a fast analysis of qualitative or partial information about the movement of objects in the environment. Examples of the latter type of task include the detection of looming motion, which might indicate an object about to collide with the observer, or the detection of sudden global rotations or translations of the visual field that might indicate an unexpected observer motion that must be counteracted to maintain balance. Such tasks require fast, simple, robust routines for detecting motion discontinuities, rough 3-D shape and motion, which might provide input into reflexive mechanisms that control motor behavior underlying obstacle avoidance, postural control, or locomotion. This section addresses work by Hildreth on the use of partial information about the "time-to-collision" with an approaching surface for performing certain visual tasks.

A number of behavioral studies suggest that biological vision systems use estimates of the time-to-collision with an approaching surface, in the control of visually-guided motor behavior [for a review, see Regan, Kaufman and Lincoln, 1986]. For restricted classes of motion, an approximation of time-to-collision can be computed from simple measures on the changing image. If we let $r$ denote the size of an object in the image (assuming perspective projection), and $\dot{r}$ denote the rate of change of image size over time, then the time-to-collision with an object is given approximately by the ratio $\frac{r}{\dot{r}}$. Through theoretical analyses and computer simulations, Hildreth has examined the validity of this simple approximation to time-to-collision, and the use of this measure for two visual tasks: (1) the recovery of the 3-D trajectory of moving objects from their 2-D projection onto the image, and (2) a simple navigation task, in which the observer must move toward a moving target while avoiding moving obstacles in the environment.

In theory, the approximation $\frac{r}{\dot{r}}$ only holds for objects that are oriented parallel to the image projection surface and undergoing pure translation toward the viewer. If, for example, an object of length $R$ at a distance $Z$ from the observer is slanted with an angle $\sigma$ from the orientation parallel to the projection surface, and is moving directly toward the observer, then the actual time-to-collision $T_c$ is given by the following expression

$$T_c = -\frac{Z}{\dot{Z}} = \left(\frac{r}{\dot{r}}\right)\left[\frac{4Z^2 + R^2\sin^2\sigma}{4Z^2 - R^2\sin^2\sigma}\right].$$

Clearly, if $Z$ is large compared to $R$, or if $\sigma$ is small, then $T_c$ can be approximated by $\frac{r}{\dot{r}}$. Other expressions can be obtained for the cases where the object is not translating directly toward the observer, or where the object is rotating as it moves. These relationships can easily be derived for the case of spherical projection as well.

Approximations to time-to-collision, which may be obtained straightforwardly from the changing image, can be combined with measures of the projected velocities of image features to derive the 3-D heading of objects in space. This 3-D heading at each moment in time can then be used to reconstruct the 3-D trajectory of a moving object, if one assumes an initial depth for the object (the trajectory scales in depth by this initial guess). Hildreth has designed a model for performing this recovery of 3-D motion, and has implemented and begun to test this model in a computer vision system. Perceptual studies are also being conducted to test the validity of this model for the human visual system.

Knowledge of 3-D heading, derived from rough time-to-collision estimates, can also be used in simple navigation tasks. Information about the 3-D heading of an object being tracked can be used to compute a desired heading for the observer in order to intercept the target. Similarly, information about the 3-D heading of obstacles can define a range of observer headings that would avoid them, and the time-to-collision estimate itself determines whether the observer should bother trying to avoid a particular obstacle. Hildreth is developing a simple simulation system that performs this navigation task in an artificial environment, using time-to-collision estimates to derive the 3-D trajectories of objects in the environment. In the future, we plan to test this model on natural imagery.

### 5.4. Using Recognition for Mobile Robot Localization

Long-term autonomy for mobile robots presents problems which are different from those encountered with the more limited mission-level autonomy of typical mobile robot systems. David Braunegg is researching the problem of building and maintaining a world model representation to support the navigation of long-term autonomous mobile robots. This research is investigating the use of stereo vision to obtain enough information about the world to enable path-planning and navigation-planning to be performed. Analysis of the requirements on a representation which can support long-term autonomy has led to the design of a two-level world model which is currently being implemented. In a related part of this research, Braunegg is developing a method for recognizing world locations based on stereo data.

### 5.5. Photogrammetry

A new method for the classic photogrammetric problem of relative orientation that arises in work on binocular stereo and motion vision has been derived by Berthold Horn [1987]. While iterative in nature, like other existing solutions, it does not require a good initial guess, and is able to select the correct solution from among several that minimize the departure from satisfaction of the coplanarity constraint equations. The new method can be implemented using unit quaternions (Euler parameters) to represent rotation, and was inspired by the closed form solution of the absolute orientation problem recently discovered.

We have also continued to study so-called "critical surfaces" that lead to difficulties in recovering the relative orientation. These arise in motion vision as well with a somewhat different interpretation.

### 5.6. Regularization and Optimal Filtering

Classic optimal filtering methods yield linear shift invariant filters (convolutional operators) that best recover signal in the least-squares sense. The filter design is based on the correlation functions or, equivalently, power spectra of signal and noise. As originally discussed by Tikhonov and others, regularization methods recently applied widely in machine vision can be interpreted in terms of such filters [see for instance Bertero et al., 1986; Geiger and Poggio, 1987]. Horn has considered the idea of "reverse engineering" an optimal filter given a particular regularization term, that is, to discover what ratio of signal power spectrum to noise power spectrum leads to an optimal filter like the one obtained using a specific regularization term. This problem is related to the choice of the optimal regularization parameter $\lambda$ discussed in our report in the last Proceedings [see also Geiger and Poggio, 1987].

As already indicated by Tikhonov (and by the Bayesian interpretation of regularization), the selection of the regularization stabilizer can be done by a systematic procedure provided that some of the statistics of the signal and the noise are known. Finally, simple regularization terms (such as the integral of the square of the *nth* derivative) lead only to a small subset of all possible convolutional operators. The optimal filter approach is not restricted in this way.

### 5.7. Surface Discontinuities from Stereo

Richard Wildes has been studying the recovery of three-dimensional surface discontinuities from a binocular disparity map. The initial investigation has restricted consideration to planar surfaces. The approach is based on an analysis of the differential imaging properties of textured surface patches in the neighborhood of a discontinuity. The formal results have been used to motivate methods which variously use horizontal, vertical and orientational disparities to recover surface discontinuities in three- space up to a relief transformation. Three empirical studies of these methods have been carried out. First, numerical studies of these methods show that the use of vertical disparities results in a system of equations which is unstable in the face of slight perturbations to the input data. Methods which make use of only horizontal and orientational disparities remain stable in the face of such noise perturbations. Second, psychophysical studies suggest that human observers do not make use of vertical disparities in this task. Third, an implementation which makes use of horizontal and orientational disparities has been developed (vertical disparities were not exploited due to the results of the numerical and psychophysical studies). The implementation has been tested with synthetic data, and has yielded positive results.

### 5.8. Implications of Visual Routines for Early Vision

Visual perception has at times been viewed as the problem of describing "what is where" in the scene. It may be more productive to regard vision as an ongoing selective readout of meaningful spatial entities and their relevant properties and relations, in a sequence governed by the immediate goals of the observer. This view respects the fact that most scenes are too complex for a single complete description – one satisfying every possible momentary requirement of the observer – to be feasible.

Shimon Ullman [Ullman 84] proposed a framework to support this more pragmatic statement of the vision problem. In his framework, spatial entities and relations are extracted by the goal-directed application of visual routines – sequences of elemental spatial operations drawn from a small fixed set – to local representations of the scene that have been computed bottom-up and in a spatially uniform manner. The basic operations from which visual routines are assembled – such as boundary tracing, region coloring, location marking, shifts of processing focus, and indexing of salient locations – constitute an "instruction set" for spatial analysis. The capacity for combining these operations in novel ways makes it possible to extract an open-ended set of abstract spatial entities, properties, and relations. Sharing of the machinery implementing these operations provides an essential computational efficiency. Visual routines are a satisfying account of the interface between low-level vision (the essentially bottom-up computation of local scene properties) and high-level vision (the recognition of meaningful spatial entities and the establishment of useful spatial relations): high-level processes like object recognition and spatial reasoning control scene analysis by selectively applying basic operations to the local representations of the scene. In Mahoney's [1987] detailed investigations of this idea, some important consequences for the organization of low-level vision have emerged which are discussed in the following paragraphs.

*Image Chunking.* At the outset, Ullman observed that the critical reliance on the basic operations implies that the implementation of these operations must be highly efficient. Thus a number of workers have explored efficient schemes for tracing, coloring, and indexing. Their results converge on the idea that low-level vision should generate a variety of representations, each specialized to support an efficient implementation of a particular basic operation; specialized representation is the key to the required efficiency. Specifically, it has been shown that tracing, coloring, and indexing operations can be made very time efficient, in the context of a simple, local model of parallel computation, by the introduction of specialized representations whose primitive elements are extended regions of the image, not points. Tracing and coloring, for example, become efficient for the simple reason that there are very few elements to be operated in comparison to the number of pixels. The process of subdividing the image in parallel into regions which may each be treated as a unit is referred to as *image chunking.*

*Local Boundary Integration.* Scene objects are defined primarily by discontinuities in image properties at their boundaries. Thus boundaries play a crucial role in the extraction of meaningful figures by visual routines; image chunking for tracing, coloring, and indexing is applied to a representation of the scene boundaries. Because of the complexity of real scenes, however, no single property can generally be expected to provide complete boundary information. Thus it is critical that partial boundary information provided by different properties somehow interact to generate a unified boundary representation to which chunking processes may be applied.

*Local Boundary Selection/Suppression.* The performance of chunk-based basic operations (e.g., tracing, coloring, etc.) has proven, in general, to depend on the geometric complexity, rather than the size, of the input. That is, the simpler the geometry of the input, the easier it is to build larger image chunks, and the larger the chunks, the fewer steps involved in performing some operation on the input. One important implication of this result for low-level vision processing prior to the computation of image chunks is that a local, parallel basis for singling out or emphasizing relevant boundaries, or suppressing irrelevant ones, would improve the performance of the basic operations and, consequently, facilitate the extraction of meaningful figures. For example, one might apply the chunking processes selectively to discontinuities in depth, color, or texture, or to intensity boundaries at a particular scale. The notions of selection and suppression are not, as they seem at first, contradictory with that of integration. There are a variety of possibilities for reconciling them, the simplest being that selection/suppression should strictly precede integration.

*Local Boundary Abstraction.* The second implication of the fact that input geometry determines the effectiveness of image chunking is that a local, parallel basis for suppressing irrelevant detail and noise in the (selected) boundaries also would improve the performance of the basic operations. Examples of noise that could be eliminated by local computations include short spurious line segments and small spurious gaps in continuous curves. High-frequency information in a boundary often constitutes irrelevant detail for certain purposes, such as determining connectivity, enclosure, or overall shape. Blurring, however, is not the only, or always the best, technique for eliminating irrelevant detail while preserving the useful information. Boundary abstraction is the problem of generating a representation of the scene boundaries which consists of an array of boundary tokens, each expressing only the information about the boundary at a particular location that is salient from the point of view of image chunking (and ultimately for extracting a scene entity for a given task by tracing, coloring, etc.) The requirements on the boundary tokens, then, are determined quite specifically by the chunking process to be supported. In the case of tracing, for example, the key information items are orientation and position; one problem, then, is to explore the range of possibilities for determining the local boundary orientation at a specified scale in a way that allows for thick or textured lines or edges (since chunking processes are specialized to basic operations, it may turn out that several abstract boundary representations, each specialized to support a particular chunking process, should be defined). The introduction of abstract boundary tokens is reminiscent of Marr's proposal [1982] for the primal sketch, in which abstract place tokens were defined for the benefit of explicit grouping processes, rather than relying exclusively on the grouping implicit in low spatial frequency representations. The difference is that the grouping is accomplished by the goal-directed application of an appropriate sequence of basic operations, not by Marr's spatially uniform, bottom-up, recursive grouping processes, for reasons mentioned earlier [see Ullman, 1984 for a thorough discussion].

### 5.9. The Full Primal Sketch

In order to support processes such as recognition under general conditions, it is important to extract as much relevant information about the shape of an object as possible. Eric Saund has been developing a new method for deriving a Full Primal Sketch from an image. Rather than using numerical smoothing methods, such as region smoothing using isotropic operators (such as the Gaussian), non-isotropic operators (such as Gabor filters), or contour smoothing techniques, Saund has developed a method for symbolically grouping simple tokens across scales. This symbolic scale space representation has proved very useful for preserving symbolic tokens capturing relevant information at different scales. By using techniques for dimensionality reduction to extract relevant parameters, Saund has been able to derive schemes for building more complex symbolic tokens, representing higher order shape attributes, such as corners, bars and blobs, from more primitive tokens. The representations computed in this manner seem to have several advantages over other more numerically based schemes, such as Brady's Smooth Local Symmetries [Brady and Asada, 1983], or Blum's Medial Axis Transform [Blum, 1967]. The method is being applied to the recognition of shapes from large libraries, and early results appear very promising.

### 5.10. Issues in Learning

The problem of estimating an input-output mapping from examples has taken on a new interest recently because of the peculiar excitement surrounding so-called neural nets and backpropagation algorithms that "learn". Tomaso Poggio has considered the problem of learning smooth mappings from the point of view of classical approximation and estimation theories, including polynomial and spline based estimators.

The problem of learning a mapping between an input and an output space is essentially equivalent to the problem of synthesizing an associative memory that re-

trieves the appropriate output when presented with the input and *generalizes* when presented with new inputs. It is also equivalent to the problem of estimating the system that transforms inputs into outputs given a set of examples of input-output pairs. A classical framework for this problem is *approximation theory*. Related methods are *system identification techniques* (Volterra, Wiener, etc.), used when it is possible to choose the input set, and *system estimation techniques*, used when the input-output pairs are given.

Approximation theory deals with the problem of approximating or interpolating a continuous function $f(X)$ by an approximating function $F(W,X)$ having a fixed number of parameters $W$ ($X$ and $W$ are real vectors $X = x_1, x_2, ..., x_n$ and $W = w_1, w_2, ..., w_m$). For a particular choice of $F$, the problem is then to find the set of parameters $W$ that provides the best possible approximation of $f$. This is the *learning* step. Needless to say, it is very important to choose an approximating function $F$ that is as compatible as possible with $f$. There would be little point in trying to learn an approximation if the chosen approximation function $F(W,X)$ could give only a very poor representation of $f(X)$, even with optimal parameter values.

We have suggested that one may usefully consider the problem of learning, as discussed by connectionists, as a problem of approximation, in particular of *hypersurface reconstruction* [Poggio, 1988]. Interesting connections with splines, regularization and Bayesian approaches can be immediately established.

### 5.10.1 Learning an Input-Output Mapping as Hypersurface Reconstruction

From the point of view of learning as continuous approximation – clearly only one of the several facets of learning – we can draw an equivalence with a standard approximation problem, surface reconstruction from sparse data points. Learning simply means collecting the *examples*, i.e., the input coordinates $x_i, y_i$ and the corresponding output values at those locations (the height of the surface) $d_i$. This builds a look-up table. *Generalization* means estimating $d$ in locations $x, y$ where there are no examples, i.e. no data. This requires interpolating or, more generally, approximating the surface between the data points. Interpolation is the limit of approximation when there is no noise in the data. This example, given for a surface, i.e., the graph in $R^2 \times R$, corresponding to the mapping from $R^2$ to $R$, can be immediately extended to mappings from $R^n$ to $R^m$ (and graphs in $R^n \times R^m$).

### 5.10.2 Learning and Generalization

Of course any reconstruction (or approximation) problem of this type is ill-posed in the sense that the information in the data is not sufficient to reconstruct uniquely the mapping in regions where data are not available. In addition, the data are usually noisy. *A priori* assumptions are needed about the mapping. Generalization is not possible if the mapping is completely *random* or *local*. For instance, knowing examples of the mapping represented by a telephone directory (people's names onto telephone numbers) does not help in estimating the telephone number corresponding to a new name. Generalization is based on the fact that the world in which we live is usually – at the appropriate level of description – redundant. In particular, it may be *smooth*: small changes in the inputs determine a correspondingly small change in the output (it may be necessary in some cases to accept *piecewise smoothness*). This is the most general constraint that makes possible approximation, and thus this very simple form of generalization. Other *a priori* constraints may be known before approximating a mapping, for instance that the mapping is linear [see Hurlbert and Poggio, 1987], or has a positive range, or a limited domain, or is invariant to some group of transformations. Smoothness of a function corresponds to the function being not fully local: the value at one point depends on other values nearby. This means that the input coordinates must have been chosen appropriately. The problem of choosing the appropriate input coordinates – the dimensions – is the key problem in learning.

Consider again the point of smoothness as the basis for generalization. It is possible to formulate this aspect of the learning problem in the framework of Bayesian estimation, using the MRF machinery. The *prior* distribution expresses smoothness, that is, the possibility of generalizing. If the observation model represents Gaussian noise, the resulting MAP solution corresponds to generalized splines, since it correspond to standard regularization techniques [Poggio et.al., 1985]. The prior probability distribution may, however, also represent a more specific *a priori* knowledge than just smoothness. Piecewise constancy, for instance, could be used for classification tasks (clustering is equivalent to the class of mappings that enforce the rule "nearby inputs produce nearby outputs"). Positivity, convexity, and local behaviors of various types may be captured by an appropriate prior distribution. In addition, coupled MRFs allow even more flexibility in the underlying *generalization conditions* in terms of *piecewise smoothness* by

using the *line process* [see Geman and Geman, 1984; Marroquin et.al., 1987).

### 5.10.3. Splines, Regularization and Learning

Consider the inverse problem of finding the hypersurface values $z$, given sparse data $d$. Standard regularization suggests a variational problem by writing a cost functional consisting of two terms. The first term measures the distance between the data and the desired solution $z$; the second term measures the cost associated with a functional of the solution $\|Pz\|^2$ that embeds the *a priori* information on $z$. Let us consider as a simple choice $P = \nabla_n$; thus, the problem is reduced to finding the hypersurface $z$ that minimizes

$$\|z_i - d_i\|^2 + \lambda \|\nabla_n z\|^2$$

where $i$ is a collective index representing the points in feature space where data are available and $\lambda$, the regularization parameter, controls the compromise between the degree of smoothness of the solution and its closeness to the data. Therefore $\lambda$ is directly related to the degree of generalization that is enforced. Higher smoothness is enforced in terms of minimization of

$$\|z_i - d_i\|^2 + \lambda \|\nabla_n^2 \nabla_n z\|^2.$$

The corresponding Euler-Lagrange equations are, respectively,

$$\lambda \nabla_n^2 z = d$$

and

$$\lambda \nabla_n^2 \nabla_n^2 \nabla_n^2 z = d,$$

where $\nabla_n^2$ is the obvious $n$-dimensional extension of the usual $\nabla^2$ operator.

Tomaso Poggio and Woodward Yang are presently studying the possibility of using analog networks to compute $n$-dimensional spline approximations. This is especially attractive for multidimensional splines. We have previously shown that standard regularization (SR) can be implemented using analog networks of resistors and batteries. SR is equivalent to generalized splines, and therefore generalized splines can be computed with the same type of analog networks used for 2-D surface reconstruction, but with higher connectivity.

## READING LIST


Asada, H. and M. Brady, "The Curvature Primal Sketch," *Artificial Intelligence Laboratory Memo 758*, Massachusetts Institute of Technology, Cambridge, MA, 1984.

Blum, H. "A Transformation for Extracting New Descriptions of Shape," In: **Models for the Perception of Speech and Visual Form**, W. Dunn (ed.), The MIT Press, Cambridge, MA, 362-380, 1967.

Brady, M. and H. Asada, "Smoothed Local Symmetries and Their Implementation," In: **The First International Symposium on Robotics Research**, M. Brady and R. Paul (eds.), The MIT Press, Cambridge, MA, 1983.

Brooks, M.J. and B.K.P. Horn. "Shape and Source from Shading," *Artificial Intelligence Laboratory Memo 720*, Massachusetts Institute of Technology, Cambridge, MA, 1985.

Brooks, R.A. "A Robust Layered Control System for a Mobile Robot," *IEEE J. Robotics and Automation*, **RA-2**, 14-23, 1986.

Brooks, R.A., A.M. Flynn and T. Marill. "Self Calibration of Motion and Stereo Vision for Mobile Robot Navigation," these Proceedings, 1988.

Brooks, R.A., A.M. Flynn and T. Marill. "Self Calibration of Motion and Stereo Vision for Mobile Robot Navigation," *Artificial Intelligence Laboratory Memo 984*, Massachusetts Institute of Technology, Cambridge, MA, 1987.

Bülthoff, H.H. and H.A. Mallot. "Interaction of Different Modules in Depth Perception: Stereo and Shading," *Artificial Intelligence Laboratory Memo 965/Center for Biological Information Processing Paper 024*, Massachusetts Institute of Technology, Cambridge, MA, 1987.

Bülthoff H. and H. Mallot. "Interaction of Different Modules in Depth Perception," *Proc. of the First Internat. Conf. on Computer Vision*, IEEE Computer Society, Washington, DC, 295-305, 1987.

Cass, T.A. "A Robust Parallel Implementation of 2D Model Recognition," submitted.

Cass, T.A. "Robust Parallel Computation 2D Model-Based Recognition," 1988, these Proceedings.

Gamble, E.B. and T. Poggio. "Visual Integration and Detection of Discontinuities: The Key Role of Intensity Edges," *Artificial Intelligence Laboratory Memo 970/Center for Biological Information Processing Paper 027*, Massachusetts Institute of Technology, Cambridge, MA, 1987.

Geiger, D. and T. Poggio. "An Optimal Scale for Edge Detection," *Proc. of the Intl. Joint Conference on Artificial Intell.*, 745-748, Milan, August, 1987.

Geiger, D. and T. Poggio. "Level Crossings and the Panum Area," *Proc. IEEE Computer Society Workshop on Computer Vision*, December, 1987.

Geman, S. and D. Geman. "Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images," *IEEE Trans. Pattern Matching and Machine Intell.*, **6**, 1984.

Gennert, M.A. and S. Negahdaripour. "Relaxing the Brightness Constancy Assumption in Computing Optical Flow," *Artificial Intelligence Laboratory Memo 975*, Massachusetts Institute of Technology, Cambridge, MA, 1987.

Grimson, W.E.L. "On the Recognition of Curved Objects," *Artificial Intelligence Laboratory Memo 983*, Massachusetts Institute of Technology, Cambridge, MA, 1987.

Grzywacz, N.M. and A.L. Yuille. "Massively Parallel Implementations of Theories for Apparent Motion," *Artificial Intelligence Laboratory Memo 888/Center for Biological Information Processing Paper 016*, Massachusetts Institute of Technology, Cambridge, MA, 1987.

Grzywacz, N.M. and E.C. Hildreth. "The Incremental Rigidity Scheme for Recovering Structure from Motion: Position-Based versus Velocity-Based Formulations," *J. Optical Soc. Amer. A*, **4**, 503-518, 1987.

Grzywacz, N.M., E.C. Hildreth, V.K. Inada and E.H. Adelson. "The Temporal Integration of 3-D Structure from Motion: A Computational and Psychophysical Study," *Brain Theory: Proc. of the Second Trieste Meeting on Brain Theory*, 1987, in press.

Hildreth, E.C. and C. Koch. "The Analysis of Visual Motion: From Computational Theory to Neuronal Mechanisms," *Annual Rev. of Neuroscience*, **10**, 477-533, 1987.

Hildreth, E.C. "Computations Underlying the Measurement of Visual Motion," In: **Image Understanding 1985-86**, W. Richards and S. Ullman (eds.), Ablex Publishing Co., NJ, 99-146, 1987.

Hildreth, E.C "Computational Studies of Visual Motion Analysis," In: **Models of Visual Perception: From Natural to Artificial**, M. Imbert (ed.), Oxford University Press, 1987, in press.

Hildreth, E.C. "The Computational Study of Vision," In: **Advances in Physiological Research**, H. McLennan (ed.), Plenum Press, 1987, in press.

Hildreth, E.C. "Computational Studies of the Extraction of Visual Spatial Information from Binocular and Motion Cues," *Canadian Journal of Physiology and Pharmacology*, 1987, in press.

Hildreth, E.C. and J.M. Hollerbach. "The Computational Approach to Vision and Motor Control," In: **Handbook of Physiology**, F. Plum (ed.), Americal Physiological Society, 605-642, 1987.

Hurlbert, A. and T. Poggio. "Making Machines (and AI) See," *Daedalus: Proc. Amer. Acad. Arts and Sciences*, **117(1)**, 213-239, 1988.

Hurlbert, A. and T. Poggio. "Learning a Color Algorithm from Examples," *Artificial Intelligence Laboratory Memo 909/Center for Biological Information Processing Paper 25*, Massachusetts Institute of Technology, Cambridge, MA, 1987.

Hurlbert, A. and T. Poggio. "Learning a Color Algorithm from Examples," *Science*, 1988, in press.

Horn, B.K.P.. "Relative Orientation," *Artificial Intelligence Laboratory Memo 994*, Massachusetts Institute of Technology, Cambridge, MA, 1987. Also in these Proceedings.

Huttenlocher, D.P. "Recognizing Rigid Objects by Aligning Them with an Image," *Artificial Intelligence Laboratory Memo 937*, Massachusetts Institute of Technology, Cambridge, MA, 1987.

Huttenlocher, D.P. "Recognizing Solid Objects by Alignment," 1988, these Proceedings.

Huttenlocher, D.P. and S. Ullman. "Three-Dimensional Recognition of Solid Objects by Alignment with a Two-Dimensional Image," *Artificial Intelligence Laboratory Memo 938*, Massachusetts Institute of Technology, Cambridge, MA, 1987.

Mahoney, J.V. "Image Chunking: Defining Spatial Building Blocks for Scene Analysis," Master Thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, August, 1987. Published

as *Artificial Intelligence Laboratory Technical Report 980*, 1987.

Marr, D. **Vision: A Computational Investigation into the Human Representation and Processing of Visual Information**, W.H. Freeman and Co., San Francisco, CA, 1982.

Marroquin, J., S. Mitter and T. Poggio. "Probabilistic Slution of Ill-Posed Problems in Computational Vision," *Artificial Intelligence Laboratory Memo 897*, Massachusetts Institute of Technology, Cambridge, MA, 1987.

Poggio, T. "On Approximation of Mappings and Learning," 1988, in preparation.

Poggio, T., et.al. "The Vision Machine," these Proceedings, 1988.

Poggio, T, V. Torre and C. Koch, "Computer Vision and Regularization Theory," *Nature*, **317**, 314-319, 1985.

Regan, D.M., L. Kaufman and J. Lincoln. In: **Handbook of Perception and Human Performance**, K.R. Boff et.al. (ed.), Wiley, New York, 19/1-19/46, 1986.

Voorhees, H. and T. Poggio. "Detecting Textons and Texture Boundaries in Natural Images," *Proc. of the First International Conference on Computer Vision*, Computer Society Press, Washington, DC, 250-258, 1987.

Ullman, S. **The Interpretation of Visual Motion**, The MIT Press, Cambridge, MA, 1979.

Ullman, S. "Visual routines," *Cognition*, **18**, 1984.

Ullman, S. "The Optical Flow of Planar Surfaces," *Artificial Intelligence Laboratory Memo 870*, Massachusetts Institute of Technology, Cambridge, MA, 1985.

Ullman, S. "Maximizing Rigidity: The Incremental Recovery of 3-D Structure from Rigid and Rubbery Motion," *Perception*, **13**, 255-274, 1984.

Voorhees, H.L. "Finding Texture Boundaries in Images," Master's Thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1987. Published as *Artificial Intelligence Laboratory Technical Report 968*, 1987.

# USC Image Understanding Research: 1987-88[1]

R. Nevatia
Principal Investigator

Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, CA 90089-0273

## ABSTRACT

This paper summarizes the USC Image Understanding research projects and provides references to more detailed projects and provides references to more detailed papers. Our work has focussed on the topics of: mapping from aerial images, robotics vision, motion analysis for ALV, some general techniques and parallel processing.

## 1 Introduction

This paper summarizes our research projects during the past year. These proceedings contain eight other papers from our group that describe our research in more detail [1,2,3,4,5,6,7,8]. Those topics that are covered in detail in other papers are covered only very briefly in this overview.

Our research activity has focussed on the following major topics:

- Mapping from Aerial Images
- Robotics Vision
- Motion Detection for Autonomous Land Vehicle (ALV), and
- Parallel Processing

In the following we summarize our research in these areas and provide references that contain more detail.

## 2 Mapping From Aerial Images

Our g( al here is to produce high-quality symbolic maps of complex, cultural scenes from aerial image data. For sometime, we have been working with domain of large commercial airport complexes. Such scenes have a variety of features such as the transportation network (runways, taxiways and roads), buildings (terminals, hangars, etc.) and mobile objects (airplanes, trucks, cars, etc.). Our aim is to produce descriptions of the individual objects in the scene as well as an integrated description of the entire scene including the functional relationships between the parts.

Previously we have reported on our work on extraction of runways [9]. Our technique consists of hypothesizing runways by using linear segments, forming *anti-parallels* from them and then grouping the anti-parallels on the basis of continuity and collinearity. The verification of the hypotheses comes from detecting expected markings on the runways. We have applied these techniques to a variety of airport scenes with success. In recent work, we have further enhanced our verification technique. In earlier work, all processing was performed at one resolution. This resolution is not adequate to detect all markings on the runway, on the other hand it is very expensive and unnecessary to process the entire image at the highest available resolution. For the task of verification of specific features, however, we can focus on just selected parts of the image and resegment it at the needed high resolution. Thus, we have a case of the results of higher level symbolic processing guiding the low-level segmentation on a second pass. We have found this technique to be highly effective in detecting subtle marks on the runway surfaces that increase our confidence in the detection of the runways.

Detected runways are also used to guide the detection of taxiways which tend to be adjacent to each other. Verification of taxiways can be by examining their relationships to the runways. This work is in progress at this time.

In another project, funded in part by the Defense Mapping Agency (DMA), we have been developing methods for detection and description of complex building structures. We have achieved what we believe is a major success in this effort and we are able to handle buildings with wings of different heights. The shapes are restricted to being compositions of rectangles, however. The key to the method is a

[1]This research was supported, in part, by the Defense Advanced Research Projects Agency contracts DACA76-85-C-0009 and F33615-87-C-1436, order No. 3119 and monitored by the U.S. Army Engineer Topographic Laboratories and Air Force Wright Aeronautical Laboratories respectively.

technique for perceptual grouping of low-level features into meaningful high-level structures. This method is described in detail in a paper in these proceedings [1]. We expect that this technique can be generalized to work for a broad classes of objects, in aerial scenes and in other domains, and is a major focus of our current research.

We are now investigating the systems aspects of the mapping problem. We have modules to detect significant surface structures and significant 3-D structures. We are studying how these descriptions can aid each other in improving the confidence of detected structures, in detecting less prominent features and how they should be combined to give high-level, functional integrated descriptions.

# 3 Robotics Vision

Our concentration here is on description of 3-D shape and recognition of objects based on shape. We are developing methods for both surface and volume descriptions. Both methods rely on same underlying philosophical concepts - that complex shapes need to be described by decomposition into "simpler" parts and that the inter-relations of the parts are a significant aspect of the shape description. The decomposition can be carried out successively to the described level of detail. We call such descriptions *structured, heirarchical descriptions.*

For surfaces, we believe that appropriate places for segmentation are at the occluding (or jump) boundaries, slope discontinuities (folds) and "ridges". These features can, in turn, be computed from the curvature properties of surfaces and correspond to points or lines where the curvature is a local extremum or goes through a zero-crossing. Actual computation of these properties is made difficult due to the presence of noise in digital range data.

Our techniques for computing such descriptions have been presented previously [10,11].

In recent work, we have developed techniques for matching descriptions derived from two scenes. The two descriptions can be from different views of the same scene or one description can be from a complex scene and the other from a set of model descriptions of the objects known to the system. The former is needed for model-building, the latter for object recognition. We have tested our system on complex scenes of highly complex objects such as a car, a telephone, a chair and a table and obtained very satisfactory matching results. We believe that this system is a major advance over previous system in its ability to handle complex shaped objects. This method, with experimental results, is described in detail in another paper in these proceedings [2].

In a separate project, we are investigating the computation of volume descriptions, in terms of generalized cones, given only sparse and imperfect 3-D data of the scene. Previously, we have presented techniques that work on the class of objects known as *linear, straight, homogeneous generalized cylinders* (LSHGCs) [12,13]. Our current focus is on handling more general objects. We believe that the key is in finding the appropriate symmetry axes for the objects. However, in presence of surface markings and with fragmented boundaries, we also get fragmented and extraneous axes at the first level of analysis. The challenge is to connect the fragments in a meaningful by using geometrical relations between them. We are achieving some success in this effort but much remains to be done. We hope to report on this work in the near future.

# 4 Motion Detection and Analysis

We have a number of ongoing efforts in detection and analysis of moving objects, primarily in the context of supporting the DARPA Autonomous Land Vehicle (ALV) project, though the techniques are of much broader utility. This effort is being supported by our "Knowledge-based Vision Techniques" contract as part of the DARPA strategic computing program.

Of the many alternative approaches to motion analysis, we have chosen the long range or feature point methods. This approach involves extracting a set of reliable features in a sequence of images (lines, corners, contours, regions, etc.), finding the corresponding features in the sequence (i.e. by a series of image to image matching operations), and finally the computation of three-dimensional motion estimates based on the series of correspondences. We have addressed each of the problems separately and have begun to combine them into a coherent system.

In previous work, we have described a method for using more than two frames to robustly allow 3-D motion parameters [14]. In recent work, we have focussed on the following:

1. Establishing Correspondences between features in two images - Primitives we choose to match are *supersegments* which represent connected linear approximations of edge points. These supersegments are matched on the basis of similarity of shape and smoothness of displacement. We have obtained good results on real data; details are given in a paper in these proceedings [3].
2. System Integration - we have started integration of various modules in a system. This system is describe separately in [4]. Much of the work is still in designing the system architecture but we are already beginning

to get some experimental results.

3. Spatio-Temporal analysis - We have developed a method to analyze a sequence of image frames taken close apart in time, forming a spatio-temporal volume. The assumption that the frames are close allows us to put a bound on the amount of displacements for each point. Using a local process around each edge point (cutting slices at various angles along the time dimensions), we show how to compute the normal velocity (in the direction normal to the edge). As opposed to the study presented by Bolles et al [15], in which the whole volume is acquired then processed, we are able to process the sequence after very few frames, typically 5. We also explicitly model occlusion and disocclusion, which permits us to segment contours and robustly compute the true velocity field. We show results on both synthetic and real image sequences. This work is described in [5].
4. A new representation - W. Franzen in our group has suggested a new mathematical representation that simplifies the representation of a large class of motions. His representation is a generalization of the commonly used *homogeneous* representation commonly used to describe rigid transformations. However, homogeneous transformations are not constant for moving objects. Franzen suggests an augmentation of the transformation that he calls a *chronogeneous* transformation that is constant for a class of motions. It seems that this representation is highly general and should be useful for a variety of tasks including graphics animation, robot trajectory planning and motion analysis. For structure from motion problems, the formulation of problems becomes much easier and hence they should be easier to solve. The work on using this representation is only in its preliminary stages. The representation and the current status are described in [6].

# 5 Parallel Processing

As vision systems start becoming more practical, we need to be concerned with the speed of execution. It is clear that the needed speed can only be obtained by the use of highly parallel machines. The use of parallel machines for the lower levels (the iconic levels) of processing is rather straight-forward, but not so for the higher levels. Techniques of our group rely heavily on the extraction and use of symbolic descriptions and such techniques are much more difficult to parallelize. We are studying parallel implementations of such algorithms. Details of some of our work are given in [7]; following provides a brief summary.

Three parallel architectures have been investigated in the past year. These are the Meshes with Broadcast Buses, the Mesh of Meshes, and Reconfigurable VLSI Arrays. The Mesh with Broadcast Buses can be looked upon as an enhanced mesh to support sparse communication between processors. We have studied asymptotic performance improvements in using such arrays. The Mesh of Meshes is a general purpose VLSI based architecture which can provide linear speed up for a large number of image problems over a wide range of input size. An important feature of this organization is that the number of processors (which tends to be more expensive compared to memory) can be varied, and the speed up remains proportional to the number of processors. A related organization is the Mesh of Trees (MOT), which apparently has not been well studied in the context of image processing and vision. We have shown that poly log solutions to image problems can be obtained on the MOT . In fact, for most image computations the MOT can support divide and conquer more efficiently than the Pyramid, leading to superior performance on the MOT compared to the Pyramid. The Reconfigurable Mesh is a universal array which can simulate the mesh, MOT, and Pyramid computers. It has a compact VLSI layout ($O(n^2)$ for an $n \times n$ array) and the reconfiguration feature can lead to a variety of interconnection patterns among the PEs.

We have identified certain sparse data movement operations which are essential to efficient parallel solutions to many image problems. We have developed efficient techniques to implement these operations on the above organizations. Using these techniques, asymptotically superior solution times have been obtained to problems related to extracting geometric features of images. Using this, we have been able to show that an enhanced mesh is comparable to a pyramid of corresponding size. We have derived an optimal integer sorting algorithms and optimal sorting algorithms on the Mesh of Meshes. Using these results, optimal parallel solutions to problems on an $n \times n$ image can be obtained. These solutions use $p$ processors, where $p$ is in the range 1 to $n^{3/2}$. An interesting feature is that when $p = n$, the Mesh of Meshes with $n$ processors can solve all of the above image problems in the same time as an $n \times n$ two-dimensional mesh-connected computer with $n^2$ processors. Most of our solutions are simple with a small constant factor which makes them interesting from an implementation point of view. The Reconfigurable Mesh can support fast sparse data movement leading to asymptotically superior solution times to several image problems. In fact, certain techniques on the CRCW PRAM model can be directly implemented on this model. For example, several graph problems can be solved in $O(\log n)$ time which is the same time taken by the powerful CRCW shared memory model. The reconfiguration feature can be very useful in mapping image understanding algorithms, which usually have nonregular data flow.

We have studied parallel implementations of the image and stereo matching techniques developed at USC. Our parallel implementations can run on the meshes, pyramids, or reconfigurable arrays. The simplest of these implementations is on a fixed-size systolic array of size $k^{1/2} \times k^{1/2}$ which can lead to $O(k^{1/2})$ speed up. This speed up is possible by careful partitioning of the input data and allocating them to support data dependencies in the computations .

Recently, we have been studying techniques to implement basic image computations on commercially available machines such as the Hypercube and the Connection Machine. We have developed optimal parallel algorithms for image template matching on the hypercube class of machines. Our techniques lead to optimal solutions to the template matching problem using fixed memory in each of the processors.

# References


[1] Rakesh Mohan and Ramakant Nevatia. Perceptual grouping for the detection and description of structures in aerial images. In *Proceedings of the DARPA Image Understanding Workshop*, Boston, Massachusetts, April 1988.

[2] T.J. Fan, G. Medioni, and R. Nevatia. 3-d object recognition using surface descriptions. In *Proceedings of the DARPA Image Understanding Workshop*, Boston, Massachusetts, April 1988.

[3] S.L. Gazit and G. Medioni. Contour correspondences in dynamic imagery. In *Proceedings of the DARPA Image Understanding Workshop*, Boston, Massachusetts, April 1988.

[4] Keith Price and Igor Pavlin. Integration effort in knowledge-based vision techniques for the autonomous land vehicle program. In *Proceedings of the DARPA Image Understanding Workshop*, Boston, Massachusetts, April 1988.

[5] Shou-Ling Peng and Gerard Medioni. Spatio-temporal analysis of an image sequence with occlusion. In *Proceedings of the DARPA Image Understanding Workshop*, Boston, Massachusetts, April 1988.

[6] Wolfgang O. Franzen. Natural representation of motion in space-time. In *Proceedings of the DARPA Image Understanding Workshop*, Boston, Massachusetts, April 1988.

[7] V.K.P. Kumar. Parallel architectures for image processing and vision. In *Proceedings of the DARPA Image Understanding Workshop*, Boston, Massachusetts, April 1988.

[8] Philippe Saint-Marc and Gerard Medioni. An adaptive filtering scheme for meaningful features extraction. In *Proceedings of the DARPA Image Understanding Workshop*, Boston, Massachusetts, April 1988.

[9] A. Huertas, B. Cole, and R. Nevatia. Detecting runways in aerial images. In *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, California, February 1987.

[10] T.J. Fan, G. Medioni, and R. Nevatia. Surface segmentation and description from curvature features. In *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, California, February 1987.

[11] T.J. Fan, G. Medioni, and R. Nevatia. Segmented descriptions of 3-d surfaces. accepted for publication in IEEE Journal of Robotics and Automation, 1987.

[12] Kashipati G. Rao and R. Nevatia. From sparse 3-d data directly to volumetric shape descriptions. In *Proceedings of the DARPA Image Understanding Workshop*, pages 360–369, Los Angeles, California, February 1987.

[13] Kashipati G. Rao and R. Nevatia. Computing volume descriptions from sparse 3-d data. accepted for publication in the International Journal of Computer Vision, 1987.

[14] H. Shariat and K. Price. Results of motion estimation with more than two frames. In *Proceedings of the DARPA Image Understanding Workshop*, pages 694–703, Los Angeles, California, 1987.

[15] Robert C. Bolles, Harlyn Baker, and David H. Marimont. Epipolar-plane image analysis: an approach to determining structure from motion. *International Journal of Computer Vision*, 1:7–55, 1987.

# IMAGE UNDERSTANDING: INTELLIGENT SYSTEMS *

Thomas O. Binford

Robotics Laboratory, Computer Science Department,
Stanford University, Stanford, CA 94305, U.S.A.

## Abstract

This report summarizes progress in model-based vision, implemented in the SUCCESSOR system. We present results in three areas. The first area is representation and geometric modeling. New generic modeling of object classes was achieved, models of three classes were built, and a generic model of a hallway was used to build a model of the hallway of our building. A general "painted-surface model" was integrated in SUCCESSOR for metals and non-metals, specular and non-specular reflection. A new physical model for diffuse reflection has been incorporated. Curved-axis solids of revolution (CSR) and set operations with CSRs have been added. Boundary representation by "trimmed-surfaces" has been added.

The second area is interpretation of scenes and data. Initial results were obtained in expressing structural relations, measurement evidence, and predictive evidence in a Bayesian network model. Computational complexity was addressed by basing generation of hypotheses on quasi-invariant observables. A new generic observability model for diffuse reflection from surfaces was obtained. New results were obtained on differential geometry of generalized cylinders and ribbons.

The third area is segmentation and building structured descriptions of scenes. New results are described from a hierarchical stereo correspondence system which builds descriptions of objects and surfaces. Stereo reconstruction of curved surfaces is shown. Preliminary results on texture segmentation are presented. Edge segmentation into regions is described, along with global color region segmentation and local color segmentation of edges.

*This research was supported in part by a subcontract to Advanced Decision Systems, 81093 S-1 (Phase II), "Knowledge Based Vision Task B," from a contract to the Defense Advanced Research Projects Agency. Partial support was provided by the Air Force Office of Scientific Research under contract F33615-85-C-5106

## INTRODUCTION

SUCCESSOR is intended to be a general, model-based vision system. To many, the phrase is a contradiction in terms. The key to generality is to capture general constraints. With weak representation, only specific constraints can be implemented. We concentrate on generality of representation. We aim that SUCCESSOR be a comprehensive system and that it be powerful for applications. For applications, representation of domain-specific detail is important. Our impression is that some domain-specific information can be implemented with weak representation methods, but that strong representation methods provide an opportunity fo making efficient use of detailed domain-specific information.

We do fundamental research on components and on their integration. We also integrate results of research in an experimental system for our own research and to support others in the IU community, especially for target recognition.

Our objective is to identify similarity among members of an object class, not just to match identical individuals. We aim to do it with methods which have low computational complexity without sacrificing generality, a concern which has been central throughout our work, e.g. [Nevatia and Binford 73].

We have made a substantial new step toward implementing general constraints by generic, symbolic representation of object classes and general knowledge [Kriegman, Binford, Sumaneweera 88]. We have designed and implemented an abstract, generic modeling capability in SUCCESSOR, implemented three classes of generic models and applied a generic building model to vision for a mobile robot. Generic modeling is based on an object-oriented subsystem for a few geometric types. Figure 1 shows initial success in using a generic model of the class of buildings in building a model of halls and doors in our building from the generic model, using images from our mobile robot, integrating stereo and monocular information. Figure 2 illustrates a generic model of the class of screws. Screw types were chosen to cover classes from a machinery handbook. Figure 2 also shows the generic model of the hallway. Generic display methods provide views of typical examples from object classes.

Two challenges in generality are in using different sensors, optical images, depth data, and radar data now that high resolution radar data are available, and integrating evidence from multiple sensors. New results on interpretation contribute strongly to integrated interpretation [Binford, Levitt, Mann 87]. We demonstrated preliminary results aimed toward generic interpretation by expressing structural relations, predictive knowledge from generic models, and measurement evidence in a probability network model. Probability networks are natural methods of representing constraint structures [Pearl 86]. The approach appears promising, however we work to overcome substantial problems which are apparent, i.e. modeling relations, computations of probabilities, and control of hypothesis generation. Our initial efforts dealt with computation issues by partial instantiation, i.e. by instantiating only those hypothesis nodes for which evidence was sufficient, and by using quasi-invariants for generating hypotheses. We have used quasi-invariants widely in stereo vision [Arnold and Binford 80] and in ACRONYM [Lim, Chelberg, Cowan 84; Brooks 81].

To achieve this level of generality, we require general modeling capability which has a highly symbolic, compact form. We have introduced and incorporated a general physical model for reflectivity which covers specular and non-specular reflection for metals and non-metals. A single model appears to be adequate across the entire electromagnetic spectrum [Ponce and Healey 88]. A "painted-surface" model has been added to SUCCESSOR which models surface optical properties based on these physical models. A physical model from [Reichman 73] has been introduced which is the first non-trivial model for diffuse reflection used in computer vision.

Curved-axis solids of revolution (CSR) have been added to SUCCESSOR [Ponce and Healey 88]. Solid primitives are built up by an interactive graphic editor and assembled by a simple modeling language into trees by set operations, union, intersection, and difference. Primitives are related in a graph by affixment by geometric transformations between parts, including symbolic relations between planar faces. All relations may be general expressions with parameters, which allows motion of parts. A boundary representation by trimmed surface patches has been implemented. The adaptive algorithm for computing intersections of SHGCs (straight homogeneous generalized cylinders) by building variable resolution box trees has been extended to CSRs. An algorithm for consistent set operations despite degenerate cases was implemented. Several new rendering methods have been implemented which include large speedups.

A new generic observability model for diffuse reflection from surfaces was obtained [Binford 87]. The model determines that discontinuities of surface normal (order 1), surface reflectivity (order 0), and illumination (order 0) are visible generically, i.e. except on a set of measure zero, as discontinuities in intensity images of order 0, i.e. steps in intensity. Discontinuities in surface curvature (order 2) and slope discontinuities of reflectivity and illumination (order 1) are visible generically as slope discontinuities in image intensity (order 1).

[Ponce 88a] derives uniqueness and equivalence results for generalized cylinder representations, for SHGCs. He also compares ribbons of different classes and finds a practical local test for curved ribbons with skewed symmetry.

We describe new results with a hierarchical stereo vision system which builds monocular interpretations of surfaces and bodies. Stereo correspondence is made between surfaces and bodies to reduce ambiguity and computation [Lim and Binford 88]. The stereo system includes a solution to the long-standing problem of correspondence from limbs of featureless curved surfaces.

We also report edge segmentation of images into closed regions by curvilinear extension and determining cycles in the connectivity graph. Ribbon determination by projection has also been accomplished. The report also describes local color region discrimination with curvilinear edgel linking. [Sumaneweera et al 88]. Texture segmentation remains a great problem. Some preliminary results in texture discrimination show about 90% success in discriminating pairs of very different textures from Brodatz's collection. [Vistnes 88].

## INTERPRETATION

Our theme in interpretation is recognition by structure. Structures are 3d generalized cylinder primitives linked at joints. Measurements provide constraints on ribbons, i.e. images of generalized cylinders, and constraints on relations between them. To relate ribbons to generalized cylinders, i.e. to relate images to surfaces, we use quasi-invariants and inference rules [Binford, Levitt, Mann 87]. This leads to a network formulation of interpretation and recognition which includes geometric constraints (non-statistical) and statistical measurement uncertainty. Figure 3 shows objects from the class of plumbing fittings from that exercise. The Bayesian network probability formulation of [Pearl 86] was used to solve for overall probabilities of hypotheses in a test case generated interactively from edge segments as test data.

A key issue is control of interpretation within the network. This was addressed in two ways in that work. First, hypotheses were instantiated only based on measurement data or prior evidence. Second, quasi-invariants limited the set of hypotheses generated. The effect of these two is to change the probability network to an incomplete network. We have not yet addressed the change which results to the probability computation from partial instantiation. In [Nevatia and Binford 73], indexing based on topology of axes of generalized cylinders was used for hypothesis generation from depth data. Hypotheses were verified with statistical tests of hypotheses against size ratios of generalized cylinders.

[Binford 88] derives a generic observability model for non-specular reflection. He considers the image intensity equation and determines which discontinuities of image intensity are caused by surface discontinuities of order 0, 1, and 2 (discontinuities in depth, tangent plane, and curvature, respectively), reflectivity discontinuities of order 0, 1, and 2 and discontinuities in illumination of order 0, 1, and 2. Generic observables are observable except on a compact set of measure zero, the standard mathematical definition. The results are that surface discontinuities of order 0, 1, and 2 are generically observable. Reflectivity discontinuities of order 0 and 1 and discontinuities in illumination of order 0 and 1 are generic observables. These results have consequences for perceptually adequate representation, e.g. surface representations must be $C^2$ except at boundaries, otherwise they introduce spurious discontinuities in image intensities. The results have the consequence that it is possible to generate generic predictions about image discontinuities.

[Ponce 88a] uses differential geometry of SHGCs to prove several uniqueness results. Parabolic lines of an SHGC are either meridians or parallels, i.e. either constant z or constant $\theta$. If a surface is described by two SHGCs with the same axis and cross section plane, then the SHGCs are equivalent. A surface described by an SHGC with two parabolic meridians and two parabolic parallels has no other non-equivalent SHGC description. For a nonlinear SHGC, the direction of the axis is determined by the direction of the cross-section, and the converse. A surface with at least two parabolic lines has at most one SHGC description if these lines are parallel, and at most three if they intersect. An SHGC is not necessarily regular; necessary and sufficient conditions are

derived for regularity.

[Ponce 88b] finds a local condition for skewed symmetries for curved ribbons. It extends the angular bisector condition for straight ribbons to a condition on curvatures. He proposes an algorithm for finding them. He also compares three classes of ribbons and extends previous results in relating them.

## REPRESENTATION

## GENERIC MODELS

The generic models in SUCCESSOR extend the representation mechanisms by which generic models of object class are implemented, compared to ACRONYM. In ACRONYM, class models were implemented by constraints on number of elements and parameters, e.g. number of engines on aircraft, and range of sizes. In the new work reported here, the basis for constraints is larger. Instead of varying numbers, variants for components can be within a type structure. The underlying object-oriented subsystem has classes, sets, numbers and mappings There is a type system. Inheritance is based on subclasses, i.e. specializations, similar to ACRONYM. Curves and surfaces are represented at a level of abstraction. Geometric constraints can be instantiated as algebraic constraints on which symbolic manipulation can be performed. A generic model of generalized cylinders was implemented to provide an interface to the geometric modeling part of SUCCESSOR. This model represents generalized cylinders explicitly as a spine curve and a cross section surface. Constraints are more difficult to show in a figure than the set operations shown. About 90% of screw types in a machinery handbook have been modeled. A number of constraints do not show up in figure 2: e.g. head and shaft are coaxial: there are scaling rules, ratios of sizes, which determine how components of different sizes of a screw type scale. The result is remarkably compact and powerful.

The generic hallway model is to be part of a generic building model. The particular example was chosen in part as a simple example to demonstrate effective use of a generic model to guide perception in building a model of our particular building. Even at an early stage, initial results gave a much richer perception of the hallway than the special purpose vision system from the mobile robot. Important parts of the model were functional, not geometric. Geometric constraints are derived from the functional constraints. Major functions included were: 1. Environmental isolation generates enclosure and dictates connection of rooms to a hallway. 2. Human movement and occupancy places constraints on height and width of doors, width of hallway, and height of rooms. 3. Gravity affects construction to make walls and doors vertical and floors horizontal. 4. Cost limits the size of construction to slightly larger than human dimensions.

The next step is to use generic models in other model-based vision domains, especially the domain of pipe fittings.

## PHYSICAL MODELS OF SURFACES

Modeling of optical properties of surface materials and of sources has now been implemented in SUCCESSOR [Ponce and Healey 88]. The surface material model is based on a comprehensive model which is imported from physics. The model includes homogeneous materials like metals and inhomogeneous materials like plastic, specular and non-specular reflection. Non-specular reflection from inhomogeneous materials is important for the vast majority of surfaces seen in everyday life. A new model from [Reichman 73] is an extension of Kubelka-Munk theory which we have used for spectral properties to date. The new model has great significance in prediction and recognition of materials. Surface reflection, specularity, is modeled by Fresnel reflection augmented by the analysis of [Torrance and Sparrow 67]. Body reflection is given by Reichman's major extension of the Kubelka-Munk theory. This is the first non-trivial model for diffuse reflection from the body of an inhomogeneous material

in vision or graphics.

The model has some immediate consequences. It allows discrimination of metals from nonmetals from color images [Healey and Binford 88]. The model predicts color variation with geometry of body reflection from inhomogeneous materials, a variation overlooked so far by other treatments. Another consequence of the model is that reflection from homogeneous surfaces should have the same spectral response over the surface, whether at a specularity or far from one, over an intensity range of several orders of magnitude. Figure 4 shows results of experiments carried out which confirmed this prediction. We are unaware of any such previous result.

## GEOMETRIC MODELS OF SURFACES

Geometric models in SUCCESSOR are part-whole graphs with generalized cylinders as volume primitives. The class of generalized cylinders implemented in SUCCESSOR previously included star-shaped SHGCs, i.e. straight homogeneous generalized cylinders, which have straight axis, star-shaped cross section, and arbitrary scaling along the axis. Curved-axis solids of revolution (CSR) have been added and integrated [Ponce and Healey 88]. They have circular cross section with variable radius, i.e. swept with a scaling function along an arbitrary curved axis in 3-space.

Solid primitives are built up by an interactive graphic editor and stored as files. Primitives are assembled by a simple modeling language into CSG trees (computational solid geometry) with primitives connected by set operations, union, intersection, and difference. Primitives are related in a graph by affixment relations which represent geometric transformations between coordinate systems of the parts. Geometric relations between parts may be specified by symbolic geometric relations between planar faces of primitives, e.g. "with face ¡face1¿ in contact with face ¡face2¿". These relations and other affixment relations may be general expressions with parameters such as time. Thus, articulated objects which move in time are routinely modeled.

A boundary representation by trimmed surface patches has been implemented. Trimmed surface patches are subsets of the 2-d parameter space of the surface, limited by intersections and other restrictions. The adaptive algorithm for computing intersections of SHGCs by building variable resolution box trees has been extended to CSRs (curved-axis solids of revolution). Intersections are represented by polygons in the parameter space of generalized cylinder primitives. To maintain consistent representation despite degenerate cases, an analysis of [Ladlaw, Trumbore, and Hughes 86] was implemented. The intersection algorithm is of order $\sqrt{n}$ in the number of polyhedral faces necessary to approximate the surfaces uniformly to the desired resolution.

Cross sections and axes of CSRs are formed of curve segments which are now straight lines and cubic splines, joined at knots which may be $C^0, C^1, or C^2$, i.e. with 0, 1, or 2 continuous derivatives.

Several new rendering methods have been implemented. Previously, limbs of SHGCs and tubes were calculated in closed-form analytic solutions. Z-buffer methods and ray tracing were implemented for hidden surface display. Now, back-to-front facet painting for single objects has been implemented. The method is quite efficient. Ray tracing is performed with the quadtree representation of surface patches. Also, the screen is organized as a quadtree. Some speedups have been incorporated for ray tracing for line drawing display. Ray tracing is performed only at limbs of surfaces. Figure 5 shows an example. Further, visibility of contours changes only at occlusions, i.e. T-junctions. This has been implemented.

## SEGMENTATION

[Lim and Binford 88a] show results from an algorithm which presents a solution to the long-standing problem of stereo surface reconstruction from two views of a featureless curved sur-

face. Figure 6 shows the problem, two views of the limb of a curved surface do not correspond. The limb depends on viewpoint. The solution is to make accurate constraints, i.e. without surface marks, the constraint is that we have four rays which are tangent to the surface on an epipolar plane. These are four constraints. Surface marks for which correspondences are established provide additional constraints. One choice for cross section curve is a conic, which has five parameters. Four tangents alone have only four constraints. Adding a constraint which minimizes perimeter to area provides a unique solution. There may be other evidence in an image, e.g. edges which terminate the surface. By assuming a generalized cylinder which scales along the image, the terminating edge puts additional constraints on the curved surface estimate from the image. Figure 7 shows a result for a cylinder.

[Lim and Binford 88b] show new results from a hierarchical stereo system which builds high level interpretations from separate monocular images. It organizes the image into extended edges which are organized into vertices [Nalwa and Binford 86, Nalwa and Pauchon 87]. The accuracy of vertex determination is improved by extrapolating edges into vertices. Figure 8 shows this organization. Three-dimensional interpretation enables grouping extended edges into surfaces, accounting for occlusion. Surfaces are grouped into bodies. Figure 9 shows left and right images of a scene grouped into bodies. Stereo correspondence is made at the level of bodies, rather than at the level of epipolar edge elements, which is typical. There are about two orders of magnitude fewer bodies than edgels. Correspondence of bodies requires about four orders of magnitude less computation than matching of edgels.

An algorithm which finds regions as cycles of curvilinear elements has been implemented. First, edge elements are extended over gaps by curvilinearity. Intersecting curvilinear elements are determined. Cycles are determined in the connectivity graph of intersecting edges. Preliminary results are shown in figure 10. Another algorithm finds ribbons using smooth local symmetry [Sumaneweera et al 88]. It uses a projection method to cut computational complexity. Local segmentation along edges using normalized color has been demonstrated.

## TEXTURE

Currently, edge segmentation works passably for scenes with large textureless image regions. There is still room for improvement for textureless scenes, but segmentation is weak for scenes with texture. [Vistnes 88] reports progress in discriminating textures by some functions which have been believed to be important. They do seem effective in discrimination.

A dense set of directional operators were applied at a set of lengths, elongations, and directions to estimate local length, elongation, direction, and intensity [Vistnes 88]. Two types of operators were employed, much as in the visual cortex of animals, namely elongated center-surround operators and edge operators. For each individual operator, the significance of a discontinuity in the mean value was estimated by summing along strips along the operator's direction and testing the significance of a discontinuity in value of two linear fits, one on either side of the center. Figure 11 shows the form of the operator.

Overall significance was estimated by assuming that each operator was independent. A four-forced-choice experiment tested discrimation between all pairs of 10 images from Brodatz. This experiment was run three times for different numbers of lengths, elongations, and orientations of center-surround operators, and lengths and orientations of edge operators. Discrimination was high, above 90% for typical preferred choice of operator set. To compare with a standard texture operator, Laws operator was run on the same experiment. It achieved 40% discrimination. The textures used were quite varied and did not have small-scale texture. These results are encouraging for further research.

## REFERENCES


[Arnold and Binford 80] R.D.Arnold and T.O.Binford; @i("Geometric Constraints in Stereo Vision"); Proceedings of the SPIE Meeting, San Diego, CA, July 1980.

[Binford 87] T.O.Binford; "Generic Surface Interpretation· Observability Model"; International Symposium on Robotics Research, 1987.

[Binford, Levitt, Mann 87] T.O.Binford, T.S.Levitt, W.B.Mann; "Bayesian Inference in Model-Based Machine Vision"; AAAI Workshop on Uncertainty in AI, 1987.

[Brooks 81] R.A.Brooks; "Symbolic Reasoning among 3-d Models and 2-d Images;" A.I. Journal, 1981.

[Healey and Binford 88] G.Healey and T.O.Binford; "Predicting Material Classes"; Proc DARPA Image Understanding Workshop, 1988.

[Kriegman et al 88] D.J.Kriegman, T.O.Binford, T.Sumaneweera; "Generic Models for Robot Navigation"; Proc DARPA Image Understanding Workshop, 1988.

[Ladlaw, Trumbore, Hughes 86] D.H.Ladlaw, W.B.Trumbore, J.F.Hughes; "Constructive Solid Geometry for polyhedral objects"; Proc SIGGRAPH-86, 1986.

[Lim and Binford 88a] H.S.Lim and T.O.Binford; "Curved Surface Reconstruction Using Stereo Correspondence"; Proc DARPA Image Understanding Workshop, 1988.

[Lim and Binford 88a] H.S.Lim and T.O.Binford; "Structural Correspondence in Stereo Vision"; Proc DARPA Image Understanding Workshop, 1988.

[Lim,
Chelberg, Cowan 84] H.S.Lim, C.K.Cowan, D.M.Chelberg; "ACRONYM Model-Based Vision in the Intelligent Task Automation Project"; Proc SPIE San Diego, 1984.

[Nalwa and Binford 86] V.S.Nalwa and T.O.Binford; "On Detecting Edges"; IEEE Transactions on Pattern Analysis and Machine Intelligenece, 8, p699, 1986.

[Nalwa and Pauchon 87] V.S.Nalwa and E.Pauchon; "Edgel-aggregation and edge-description"; Computer Vision, Graphics, and Image Processing 40, p79 1987.

[Nevatia and Binford 73] R.Nevatia and T.O.Binford; "Structured Descriptions of Complex Objects"; 3rd Int Conf on AI 1973.

[Pearl 86] J. Pearl; "Fusion, Propagation, and Structuring in Belief Networks"; Artificial Intelligence 29, p241, 1986.

[Ponce and Healey 88] J. Ponce and G.Healey; "Using Generic Geometric and Physical Models for Representing Solids"; Proc DARPA Image Understanding Workshop, 1988.

[Ponce 88a] J. Ponce; "Straight Homogeneous Generalized Cylinders: Differential Geometry and Uniqueness Results"; Proc DARPA Image Understanding Workshop, 1988.

[Reichman 73] J. Reichman; "Determination of Absorption and Scattering Coefficients for Nonhomogeneous Media. 1: Theory"; Applied Optics, Vol 12, p 1811, 1973.

J. Reichman; "Determination of Absorption and Scattering Coefficients for Nonhomogeneous Media. 2: Experiment"; Applied Optics, Vol 12, p 1816, 1973.

[Sumaneweera et al 88] T.Sumaneweera, G.Healey, B.Y.Lee, T.O.Binford, J.Ponce; "Image Segmentation Using Geometrical and Physical Constraints"; Proc DARPA Image Understanding Workshop, 1988.

[Torrance and Sparrow 67] K. Torrance and E.Sparrow; "Theory for Off-Specular Reflection from Roughened Surfaces"; J. Optical Society of America, 57, p1105, 1967.

[Vistnes 88] R. Vistnes; "Texture Models and Image Measures for Segmentation"; Proc DARPA Image Understanding Workshop, 1988.

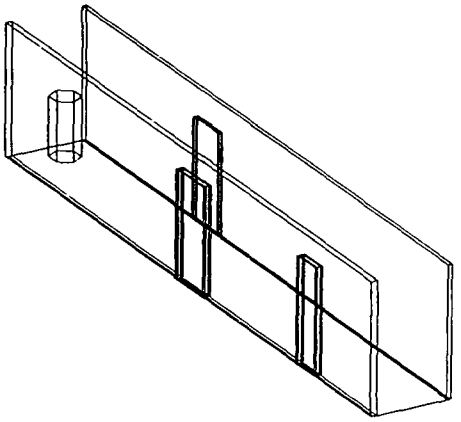


Figure 1: model of hallway built using generic hallway model with stereo and monocular edge data.

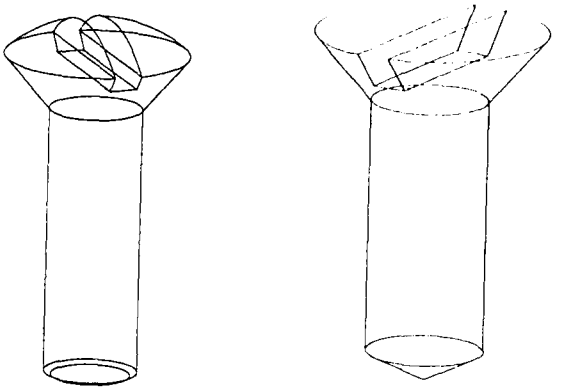


Figure 2 b) instance of screw

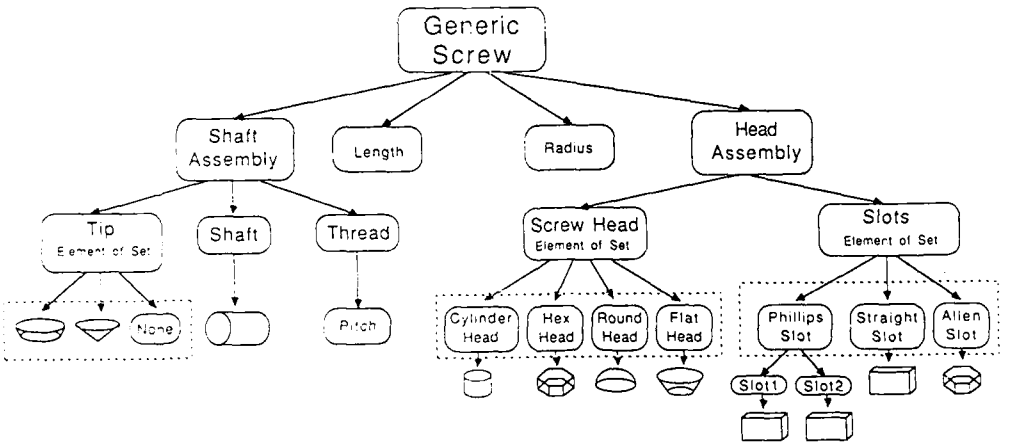



Figure 2: a) generic model of the class of screws.

# Generic Building Model

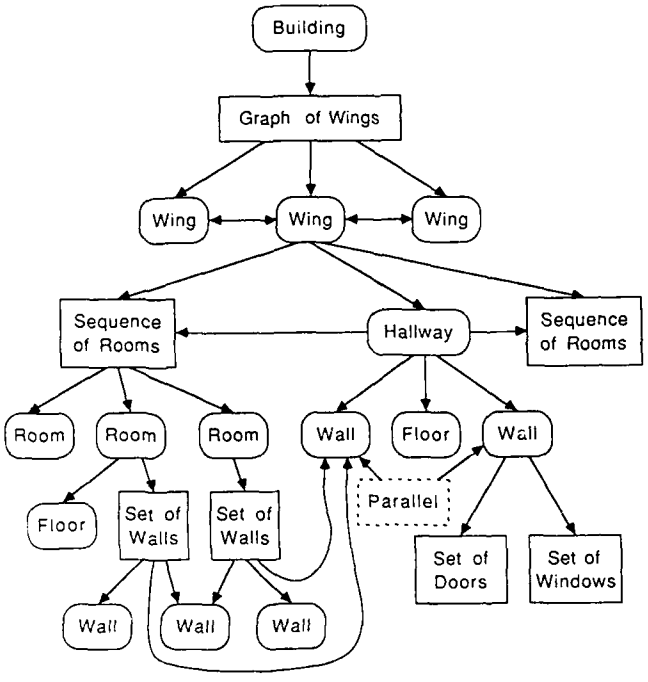



Figure 2 c) generic building model.

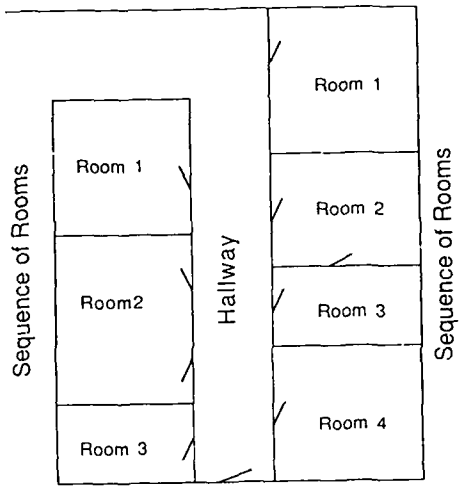



Figure 2 d) instance of building

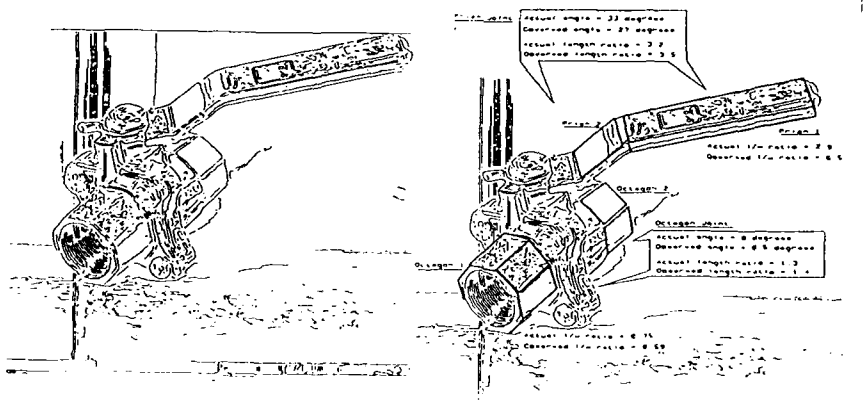


Figure 3: a) edges for valve
b) regions used in recognition

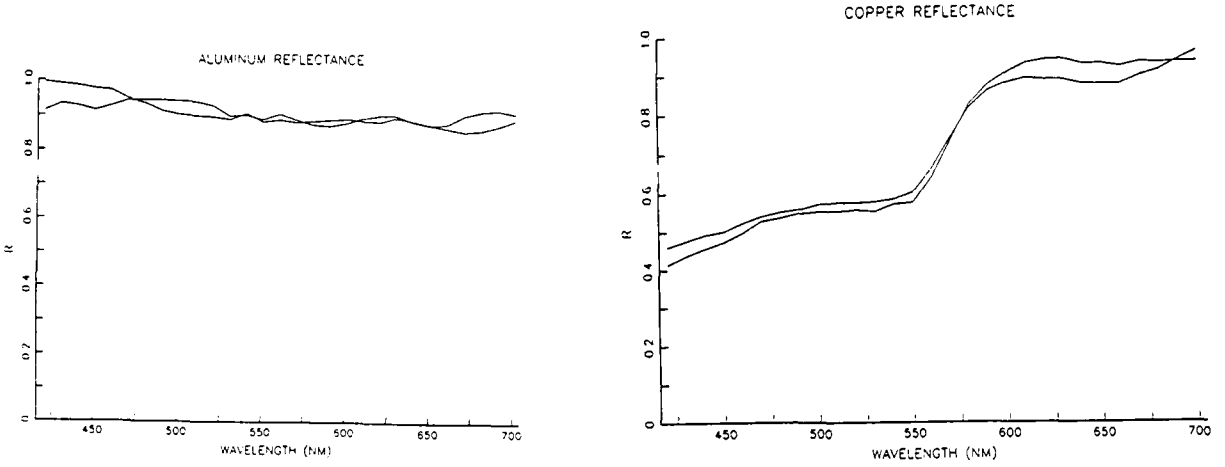



Figure 4: spectra of reflection for metals at specular peak and far from the peak a) aluminum; b) copper

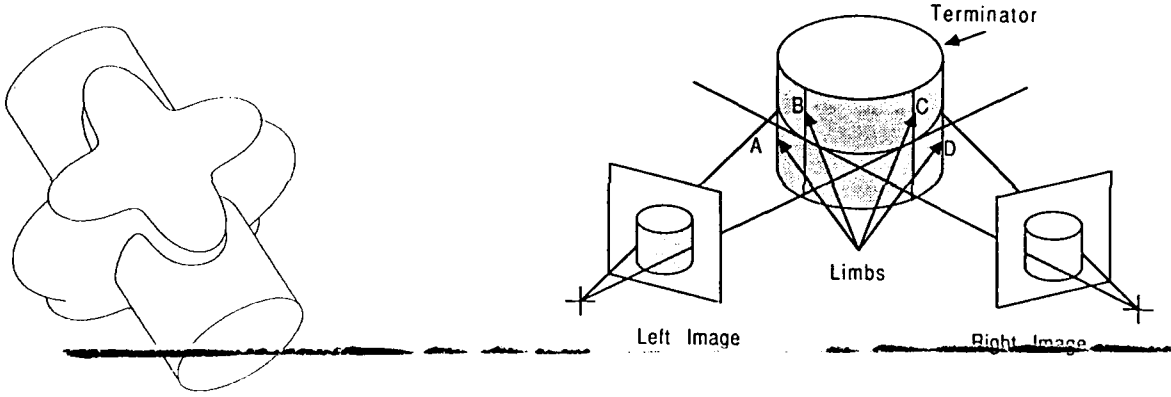



Figure 5: line drawing of curved surfaces. Ray tracing was done only at boundaries.

Figure 6: two views of the limb of a curved surface do not correspond. The limb is viewpoint dependent.

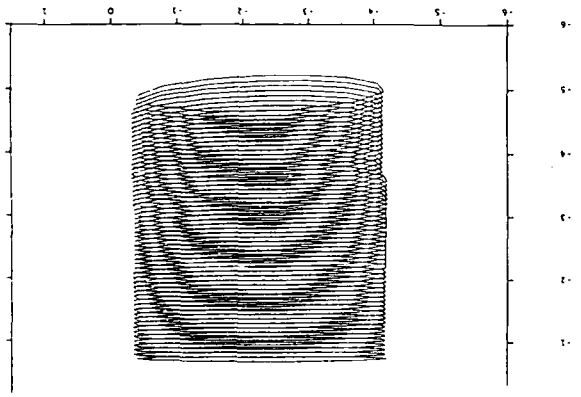


Figure 7: curved surface estimate using stereo with tangent constraints.

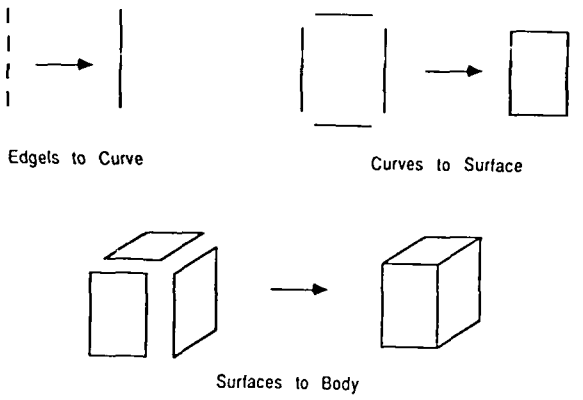
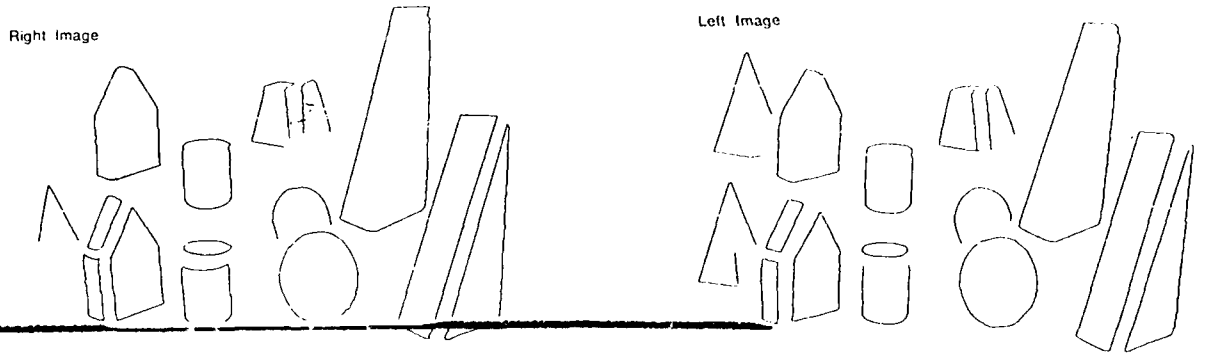



Figure 8: Hierarchical stereo using surfaces and bodies.



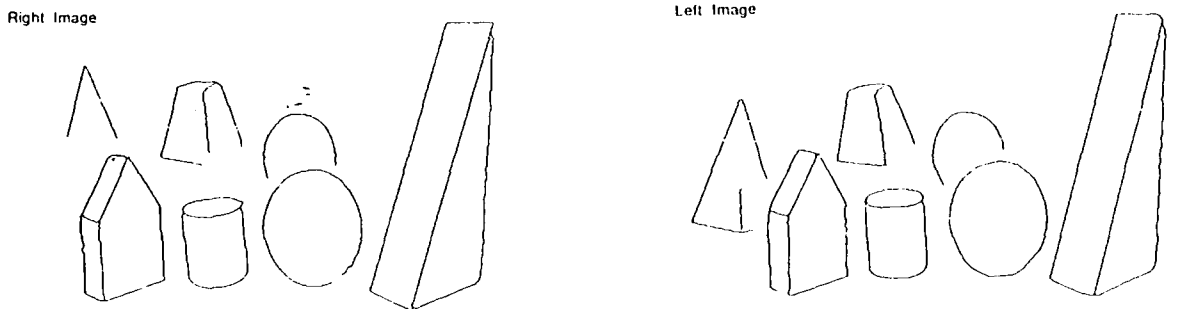




Figure 9: A stereo pair grouped into surfaces and bodies.

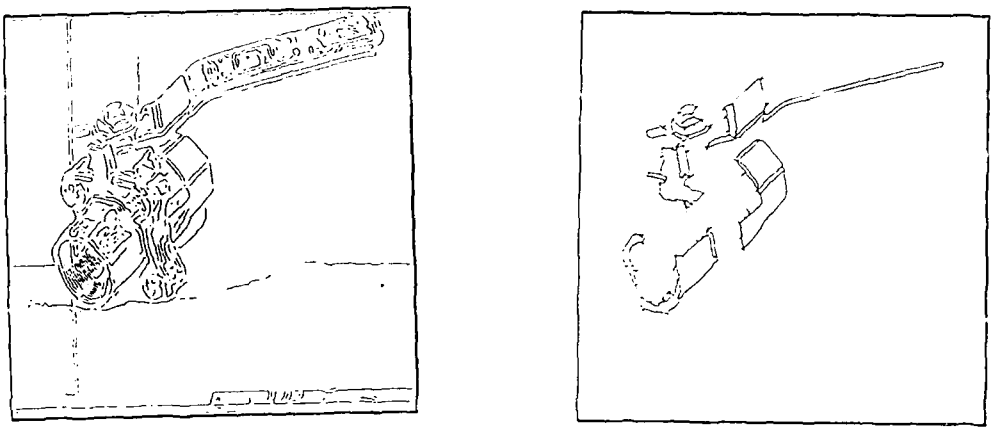
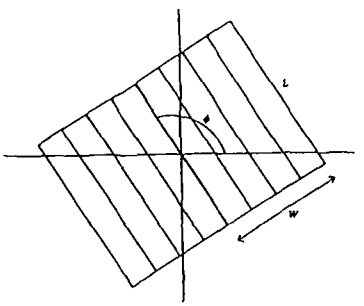
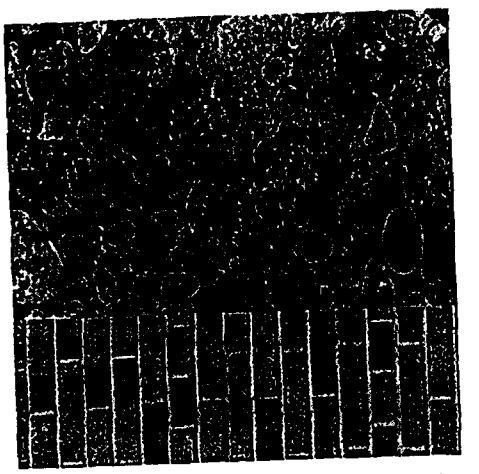


Figure 10: Cycles of edge elements.



An edge detector. It divides its receptive field into $N$ slices (here, $N = 4$) on either side of a hypothesized edge, as shown.



A texture edge formed from images D23 (pebbles) and D94 (bricks).

Figure 11: Texture discrimination

# IMAGE UNDERSTANDING RESEARCH AT THE UNIVERSITY OF MARYLAND
**(December 1986 – January 1988)**

Azriel Rosenfeld
Larry S. Davis
John (Yiannis) Aloimonos

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3411

## ABSTRACT

This report briefly summarizes research in image understanding conducted at the University of Maryland during the 14-month period December 1986 through January 1988. The areas covered include motion analysis, 3D vision, range sensing, navigation, interpretation of aerial images, digital and computational geometry, parallel algorithms, "pyramid" techniques, and other topics.

## 1. MOTION ANALYSIS

Our research on the structure from motion problem has dealt with the case of rigid motion using point correspondences (microfeatures) or line or contour correspondences (macrofeatures), as well as "correspondenceless" methods. We have also studied the nonrigid motion problem, or in general, the problem of determining the transformation parameters when an object undergoes a transformation as seen in a sequence of images. We have developed a novel method of determining an observer's motion parameters when a full ($4\pi$ steradian) image flow field is available. [Other motion-related work is described in the sections on 3D vision and navigation.]

In connection with point correspondences, we have obtained new results on how many points in how many views are necessary and sufficient to recover structure. The constraints in the cases where the velocities of the image points are known, and the positions of the image points are known with the correspondence between them established, are different and have to be studied separately. In the case of two projections of any number of points there are infinitely many solutions but if we regularize the problem we get a unique solution under certain assumptions. Finally, we have developed an algorithm for learning this particular kind of regularization. [1]

We have also developed a theory of the computation of three dimensional motion and structure from dynamic imagery, using only line correspondences. The traditional approach of corresponding microfeatures (interesting points—highlights, corners, high curvature points, etc.) has shortcomings. We have obtained a closed form solution to the motion and structure determination problem from line correspondences in three views. The theory has been compared with previous ones that are based on nonlinear equations and iterative methods. [2, 3]

In the area of "correspondenceless" techniques, a method has been developed for the recovery of the three-dimensional translation of a rigidly translating object. The novelty of the method consists of the fact that four cameras are used in order to avoid the solution of the correspondence problem. The method is immune to low levels of noise and has good behavior when the noise increases. The noise immunity is so high that even though the algorithm is intended only for translating objects, its accuracy is very high even if the object is rotating (with a small rotation) as well. [4]

Given the perspective images of an object before and after a 3D rigid motion of finite magnitude, together with the depth information of the object before motion, a method has been developed to recover the motion parameters without having to solve the correspondence problem. Let image features be defined as functionals over images containing outstanding points, line segments, or surface regions on the object. The infinitesimal variation of an image feature can be expressed as a linear constraint on the motion parameters. Thus, an infinitesimal motion can be estimated by solving a set of simultaneous linear equations. Hence, if an appropriate initial value is given, a finite motion can be recovered by iteratively applying above infinitesimal estimation; after each iteration, the image is transformed, according to the estimated motion, to get closer and closer to the target image. The appropriate initial values can be chosen finitely from a bounded parameter space which is obtained from the given images. Both synthetic data and real images have been used in experiments. [5]

Determining 3-D motion from a time-varying 2-D image is an ill-posed problem: Unless we impose additional constraints, an infinite number of solutions is possible. The usual constraint is rigidity, and most of our

previous work used it. But many naturally occurring motions are not rigid and not even piecewise rigid. A more general assumption is that the parameters (or some of the parameters) characterizing the motion are approximately (but not exactly) constant in any sufficiently small region of the image. If we know the shape of a surface we can uniquely recover the smoothest motion consistent with image data and the known structure of the object, through regularization. We have developed a general paradigm for the analysis of nonrigid motion. The variational condition we obtain includes maximizing isometry, rigidity, and planarity as special cases. If the variational condition is applied at multiple scales of resolution, it can be applied to turbulent motion. Finally, it is worth noting that this theory does not require the computation of correspondence (optic flow or discrete displacements), and it is effective in the presence of motion discontinuities. [6]

In related work, we have developed a theory for the determination of the three dimensional transformation parameters of an object from its images. The input to this process is the image intensity function and its temporal derivative. In particular, our results are:

1) If the structure of the transforming object in view is known, then the transformation parameters are determined from the solution of a linear system. Rigid motion is a special case of this theory.
2) If the structure of the object in view is not known, then both the structure and transformation parameters may be computed through a *hill climbing* or simulated annealing algorithm. [7]

In the area of optical flow analysis, we have developed a theory for determining the motion of an observer given the flow field over a full 360 degree image sphere. The method is based on the fact that the foci of expansion and contraction for an observer moving without rotation are 180 degrees opposed; and on the observation that if the flow field on the sphere is considered around three equators defining the three principal axes of rotation, then the effects of the three rotational motions decouple. The three rotational parameters can thus be determined independently by searching, in each case, for a rotational value for which the derotated equatorial flow field can be partitioned into disjoint 180 degree arcs of clockwise and counterclockwise flow. The direction of translation is obtained as a by-product of this analysis. Since this search is two dimensional in the motion parameters, it can be performed relatively efficiently. Because information is correlated over large distances, the method can be considered a pattern recognition rather than a numerical algorithm. The algorithm has been shown to be robust and relatively insensitive to noise and to missing data. Both theoretical and empirical studies of the error sensitivity have been conducted. The theoretical analysis shows that for white noise of bounded magnitude $M$, the expected error is at worst linearly proportional to $M$. Empirical tests demonstrate negligible error for perturbations of up to 20% in the input, and errors of less than 20% for perturbations of up to 200%. [8]

The fundamental assumption of almost all existing computational theories for the perception of structure from motion is that moving elements on the retina correspond projectively to identifiable moving points in three-dimensional space. Furthermore, these computational theories are based on the fundamental idea of retinal motion, i.e., they use as their input the velocity with which image points are moving (optic flow or discrete displacements). We have investigated the possibility of developing computational theories for the perception of structure from motion that are not based on the concept of the velocity of individual image elements, i.e., they do *not* use optic flow or displacements as input. **A paper on this work appears in these Proceedings.**

## 2. 3D VISION

We have been concerned with the derivation of 3D shape information from single or multiple cues available in a single image or a sequence of images.

Our research involving single cues has dealt with deriving shape from pattern or texture information. We have developed a theory for the recovery of the shape of a surface covered with small elements (texels). The theory is based on the apparent surface-pattern distortion in the image and fits the regularization paradigm, recently introduced in computer vision by Poggio et al. A mapping is defined based on the measurement of the local distortions of a repeated unknown texture pattern due to the image projection. This mapping maps an apparent shape on the image to a locus of possible surface orientations in gradient space. The analysis is done under an approximation of the perspective projection called paraperspective. The resulting algorithm has been applied to several synthetic and real images to demonstrate its performance. [9]

A central goal for visual perception is the recovery of the three-dimensional structure of the surfaces depicted in an image. Crucial information about three-dimensional structure is provided by the spatial distribution of surface markings, particularly for static monocular views: projection distorts texture geometry in a manner that depends systematically on surface shape and orientation. To isolate and measure this projective distortion in an image is to recover the three dimensional structure of the textured surface. For natural textures, we have shown that the uniform density assumption (texels are uniformly distributed) is enough to recover the orientation of a single textured plane in view, under perspective projection. Furthermore, when the texels cannot be found, the edges of the image are enough to determine shape, under a more general assumption, that the sum of the lengths of the contours on the world plane is about the same everywhere. Finally, experimental results have been obtained for synthetic and natural images. [10]

In connection with the general problem of 3D vision, we have studied an approximation of perspective projection, called paraperspective. It turns out that it is a very good approximation of perspectivity under a variety of situations, and it can be used very successfully in texture, contour and motion analysis, as well as object recognition. We have analyzed paraperspective projection, compared it with orthography and perspective, applied it to problems that have been addressed in a different way in the literature, and used it to discover invariant geometric relations that were unknown up to now. Our main contribution lies in the conclusion that very good results are obtained by applying to perspective images algorithms developed from a computational theory based on paraperspective projection. This, along with the simplicity of paraperspective and the fact that this projection leads to the discovery of perspective invariants, motivates the study of paraperspective projection in the context of image understanding. [11]

While we continue our research on the relation of shape to specific cues, we are aware of problems with passive monocular reconstruction from single cues. The theory of discontinuous regularization addresses these problems, which are basically the following.

(1) The assumptions employed are usually very strong (they are not present in a large subset of real images), and so some of the algorithms fail when applied to real images.

(2) Usually the constraints from the geometry and the physics of the problem are not enough to guarantee uniqueness of the computed parameters. In this case, strong additional assumptions about the world are used, in order to restrict the space of all solutions to a unique value.

(3) Even if no assumptions at all are used and the physical constraints are enough to guarantee uniqueness of the computed parameters, then in most cases the resulting algorithms are not robust, in the sense that if there is a slight error in the input (i.e., a small amount of noise in the image), this results in a catastrophic error in the output (computed parameters), and this is observed from experiments.

It turns out that if several available cues are combined, then the above mentioned problems disappear in most cases; the resulting algorithms compute robustly and uniquely the intrinsic parameters (shape, depth, motion, etc.).

We have developed a general mathematical theory for the unique and robust computation of intrinsic parameters. The computational aspect of the theory envisages a cooperative highly parallel implementation, bringing in information from five different sources (shading, texture, motion, contour and stereo), to resolve ambiguities and ensure uniqueness of the intrinsic parameters. [12] **A paper on this work appears in these Proceedings.**

Specific areas studied during the past year include shape from shading and motion [13], shape from contour and motion [14], and shape from shading and contour [15]. This work is briefly summarized in the following paragraphs.

Most of the basic problems in computer vision, as formulated, admit infinitely many solutions. An example of this is the shape from shading problem. But vision is full of redundancy and there are several sources of information that if combined can provide unique solutions for a problem. We have combined shading and motion to uniquely recover the light source direction and the shape of the object in view.

(1) We have developed a constraint among retinal motion displacements, local shape, and the direction of the light source. It is worth noting that this constraint does not involve the albedo of the imaged surface. This constraint is of importance in its own right, and can be used in related research on computer or human vision.

(2) We have developed a constraint between retinal displacements and local shape. Again, this constraint is important on its own, and it lies at the heart of the algorithms presented in this paper.

(3) We have formulated algorithms for the unique computation of the lighting direction and the shape of the object in view.

(4) Finally, we have obtained experimental results, using synthetic images, that test the theory.

The problem of shape from contour has also been examined. In traditional passive perception approaches, this problem has infinitely many solutions; and special assumptions or ad hoc heuristics must be employed in order to reduce the space of solutions to a unique value, and there is excellent research on this topic. An alternative approach is to consider an active observer, i.e., an observer that moves in a known way or employs some a priori knowledge which will enable a unique computation of shape. The theory we have developed shows how to recover shape from contour by utilizing invariant properties of contours in different perspective projections. Correspondence of features among images is not used. In particular, our results are: 1) A monocular observer can uniquely determine from one view the shape of a planar surface which contains multiple contours (at least two), provided the 3D area and length ratio of contours are given. 2) An active monocular observer can determine the shape of a planar contour from two views. 3) If the 3D area and length are given (model based applications), the shape of a planar contour can be determined from one view by a monocular observer. We have also obtained experimental results in this area.

We have developed an algorithm for reconstructing the shape of a cylindrical object from contour and shading without knowing the surface albedo of the object or the lighting conditions of the scene. The input image is segmented into spherical, cylindrical, or planar surfaces

by analyzing local shading. The cylindrical surface is characterized by the direction of the generating lines, determined from spatial derivatives in the image. The brightest generating line has strong constraints on the shading analysis on the cylindrical surface and leads to a simplification of the equation which represents the relation between the contour shape and the shading. Although there remains one degree of freedom between the surface normal of the base plane and the slant angle of the generating line, we can uniquely recover the cylindrical shape from this solution (up to reflection). Experimental results for a synthetic image have been obtained.

We have also studied qualitative aspects of 3D vision, based on cues such as color, shading, highlights, and transparency.

Recently many researchers have described methods for the segmentation and analysis of static scenes with shading and highlights. By taking advantage of color and the constraints inherent in the reflection model, segmentation algorithms that aspire to dealing with natural scenes have been presented. However, practical considerations and the variety of lighting situations and surface types commonplace in the natural world weaken these quantitative photometric methods. The lack of constraints in the natural world suggests strategies based on the classification of step edges, and on more robust qualitative photometric techniques. The latter resemble the types of qualitative photometry seemingly performed by the human visual system. [16]

In connection with transparency, Metelli made an important contribution by identifying order and magnitude restrictions for a pattern of intensities and showing that when they are satisfied the perception of transparency readily occurs. These restrictions were derived from a physical model of transparency. We claim that the visual system does not use intensity information to compute indices of transmittance and reflectance analogous to what an optical engineer might do in describing a physical instance of transparency. Rather, a lightness pattern affects perceptual transparency, just as geometric properties do, through processes that impose an organization on sensory information rather than through processes that recover quantitative descriptions. In the absence of depth cues, such as stereopsis and motion parallax, the perception of transparency occurs when the lightness relations in a pattern favor the perception of a continuous boundary across x-junctions. We have presented evidence for two kinds of violations of the order and magnitude restrictions, simple and strong. ansparency judgments, though reduced in number, still occur for simple violations of the order and the magnitude restrictions. Transparency judgments occur relatively infrequently for strong violations. A physical model of transparency fails to capture the difference between simple and strong violations of the order and magnitude restrictions. We have formulated (a) the basis for differentiating between simple and strong violations of the order and magnitude restrictions, (b) how simple and strong violations affect the perception of transparency, and (c) the occurrence of transparency with and without color constancy, i.e., the color seen through the transparent surface looks or fails to look the same as the color seen directly. [17]

We are also studying issues connected with the representation of 3D objects. In particular, we have developed a representation for polyhedra that does not depend on any external coordinate system. The representation contains the complete metrical as well as topological information from which a polyhedron can be reconstructed. Moreover, the representation is unique. We have also developed algorithms for finding all of the rotational and reflectional symmetries of a polyhedron using this representation. These algorithms do not perform any numerical computation, they are practical and have been implemented in Franz Lisp. [18]

A major goal of computer vision is object recognition, which involves matching of images of an object, obtained from different, unknown points of view. Since there are infinitely many points of view, one is faced with the problem of a search in a multidimensional parameter space. A related problem is the stereo reconstruction of 3-D surfaces from multiple 2-D images. We propose to solve these fundamental problems by using geometrical properties of the visible shape that are invariant to a change in the point of view. To obtain such invariants, we start from classical theories for differential and algebraic invariants not previously used in image understanding. As they stand, these theories are not directly applicable to vision. We suggest extensions and adaptation of these methods to the needs of machine vision. We study general projective transformation, which include both perspective and orthographic projections as special cases. [19] **A paper on this work appears in these Proceedings.**

## 3. RANGE SENSING

We have developed a laboratory range sensor capable of producing dense range images [20], and are using it to collect range data for a variety of applications. We have also developed algorithms for constructing 3D models of a scene based on analysis of range data [21, 22].

Our range sensor is able to produce $512 \times 512$ range images of model scenes in the laboratory. This ranging instrument, which comprises a light-emitting slit, a cylindrical lens, a step-motor controlled mirror and a CCD camera, is compact enough to be mounted on the tool plate of a robot arm. The light source itself is mounted away from this structure, and the light is brought to the slit by a flexible fiberoptic light guide. The robot arm's motion can be controlled by inputs from the range scanner, for simulation of autonomous vehicles equipped with rangers. This system is programmed to produce range images which are comparable in many respects to range images produced by laser range scanners. With this similitude of formats, software for edge detection, object recognition, dynamic path plan-

ning or data fusion with video images can be developed on range images produced by this laboratory equipment and can be easily ported to laser ranging systems.

We have developed a simple method which determines the shape of an object by projecting a stripe pattern on to it. Assuming orthographical projection as a camera model and parallel light projection of the stripe pattern, the method obtains a 2(1/2)D representation of objects by estimating surface normals from the slopes and intervals of the stripes in the image. The 2(1/2)D image is further divided into planar or singly curved surfaces by examining the distribution of the surface normals in gradient space. The error in surface orientation has been evaluated.

More generally, we have developed a method for building a 3-D world model for a mobile robot from range sensor data. The 3-D world model consists of three kinds of maps: a sensor map, a local map and a global map. A range image (sensor map) is transformed to a height map (local map) with respect to a mobile robot. First, the height map is segmented into four categories (unexplored, occluded, traversable, and obstacle regions) for obstacle detection and path planning. Next, obstacle regions are classified into artificial objects (buildings, cars, road signs, etc.) or natural objects (trees, bushes, etc.) using both the height image and video image. One drawback of the height map—the recovery of vertical planes—is overcome by the utilization of multiple height maps which include the maximum and minimum height of each point, and the number of points in the range image mapped into one point in the height map. The multiple height map is useful not only for finding vertical planes in the height map but also for segmentation of the video image. Finally, the height maps are integrated into a global map by matching geometrical properties and updating region labels. The method has been tested on a model including many objects such as trees, buildings, and cars.

## 4. NAVIGATION

In connection with the Autonomous Land Vehicle Project, we have been concerned with four main areas:

1) Development of a vision system for autonomous navigation of roads and road networks.
2) Support of Martin Marietta Aerospace, Denver, the integrating contractor on DARPA's ALV program.
3) Experimenting with the vision system developed at Maryland on the Martin Marietta ALV.
4) Development and implementation of parallel algorithms for visual navigation on the parallel computers developed under the DARPA Strategic Computing program specifically, the WARP systolic array processor, the Butterfly and the Connection Machine. [23]

We have also developed a simulation program that provides a software testbed for autonomous navigation algorithms. The program allows the user to describe a complex world built from spheres, parallelepipeds, planar surfaces, cones, and cylinders. The program simulates the movement of an Autonomous Land Vehicle and constructs video and range images based on the ALV's field of view as the vehicle moves through the world. Ground maps of the world, as perceived by the ALV, are also created. [24]

Specific areas of research related to navigation are concerned with path planning [25], road following [26, 27], and obstacle avoidance [28, 29].

A mobile robot navigates with a limited knowledge of its environment because of the restricted field of view and range of its sensors, and the occlusion of parts of the environment in any single image. Most path planning algorithms consider only free regions and obstacles in the robot's world for path planning. We have extended the classical path planning paradigm to include occluded regions. This introduces the new problem of deciding when (or whether) to employ the sensor system during the execution of the path to, potentially, reveal the occluded regions as obstacles or free space for the purpose of replanning.

In the area of road following, we have developed a new method for reconstructing the 3-D structure of road boundaries from consecutive images. First, we estimate depth information by applying a motion stereo method to consecutive images, given an estimate of the interframe motion. The relation between depth, motion and disparity is used, since the accuracy of the depth depends on the disparity range. Next, the error of the estimated road structure due to quantization errors and motion estimation errors is evaluated. Finally, a representation for road boundaries is used that makes explicit the error of the road edge location in 3-D space. Experimental results have been obtained for an input image sequence taken by the ALV simulator robot in our laboratory.

We have also worked on the design of a vision system by which an autonomous land vehicle (ALV) navigates roads. The ALV vision task consists of hypothesizing objects in a scene model and verifying these hypotheses using the vehicle's sensors. Object hypothesis generation is based on the local navigation task, an a priori road map, and the contents of the scene model. Verification of an object hypothesis involves directing the sensors toward the expected location of the object, collecting evidence in support of the object, and reasoning about the evidence. Constructing the scene model consists of building a semantic network of object frames exhibiting component, spatial, and inheritance relationships. The control structure is provided by a set of communicating production systems implementing a structured blackboard; each production system contains rules for defining the attributes of a particular class of object frame. The combination of production system and object oriented programming techniques results in a flexible control structure able to accommodate new object classes, reasoning strategies, vehicle sensors, and image analysis techniques.

In connection with obstacle detection and avoidance, we have developed algorithms which segment range images and classify regions as being navigable or unnavigable by a land vehicle. The algorithms have been applied to data collected from the laser range sensor mounted on the autonomous land vehicle. The sensitivity of the algorithms to uncertainty in the orientation of the range sensor has been studied. Experiments on sensor calibration and image enhancement have been conducted. A computer model of an autonomous land vehicle and its environment is used which provides a valuable tool for investigating many issues of navigation with range sensors. Obstacle detection algorithms are used in conjunction with the model to demonstrate a vehicle navigating itself through an obstacle strewn world to a goal location.

The practical recovery of quantitative structural information about the world from visual data has proven to be a very difficult task. In particular, the recovery of motion information which is sufficiently accurate to allow practical application of theoretical shape from motion results has so far been infeasible. Yet a large body of evidence suggests that use of motion is an extremely important process in biological vision systems. It has been suggested that qualitative visual measurements can provide powerful perceptual cues, and that practical operations can be performed on the basis of such clues without the need for a quantitative reconstruction of the world. The use of such information is termed "inexact vision". We have investigated one such approach to the analysis of visual motion. Specifically, the use of certain measures of flow field divergence were investigated as a qualitative cue for obstacle avoidance during visual navigation. We have shown that a quantity termed the *directional divergence* of the 2-D motion field can be used as a reliable indicator of the presence of obstacles in the visual field of an observer undergoing generalized rotational and translational motion. Moreover, the necessary measurements can be robustly obtained from real image sequences. A simple differential procedure for robustly extracting divergence information from image sequences which can be performed using a highly parallel, connectionist architecture has been developed. The procedure is based on the twin principles of directional separation of optical flow components and temporal accumulation of information. Experimental results show that the system responds as expected to divergence in real world image sequences, and the use of the system to navigate between obstacles is demonstrated. **A paper on this work appears in these Proceedings.**

## 5. AERIAL PHOTOINTERPRETATION

The representation and management of domain-specific knowledge is one of the major problems in constructing intelligent image understanding systems. We have developed an efficient approach to this problem which is based on the method of projections used in range searching. The efficiency of the new scheme has been demonstrated by incorporating it into the SIGMA image understanding system which uses domain-specific knowledge for interpreting aerial images of suburban housing developments. The databases of the system are organized based on the new scheme. The projections of the predictions, or hypotheses, along two perpendicular axes are used to integrate related hypotheses. With this refinement, there is a significant improvement in the performance of the system SIGMA. The application domain of the system has been broadened to more complex images which include road intersections. [30]

We have also developed a system for identifying small clouds and their shadows on aerial photographs. The system segments an image into homogeneous regions; selects bright and dark regions as cloud and shadow candidates, respectively; and finds acceptable (cloud, shadow) pairs based on consistent relative position. The system performed quite well on three portions of aerial photographs each of which contained 10–20 small clouds (out of 25–60 bright regions). Other clues could have been used to aid in cloud identification; for example, clouds have irregular shapes, do not have long straight edges, and may occlude edges or curves on the terrain. However, it was not found necessary to use such clues in the image that we used. [31]

We have done further work on a method of detecting thin curvilinear features in an image based on a detailed analysis of the local gray level patterns at each pixel. This allows operations such as thinning and gap filling to be based on more accurate information. [32] **A paper summarizing this work appears in these Proceedings.**

We have also developed a logic programming based system for interpreting aerial photographs using search techniques to find segmentations that match descriptions of imagery features. **A paper summarizing this work appears in these Proceedings.**

## 6. DIGITAL AND COMPUTATIONAL GEOMETRY

In support of our research on image understanding, we have continued to investigate a variety of problems in digital and computational geometry. We will not review this work here in detail, but will present only a brief summary of the areas studied during the past year.

We have developed methods of constructing visibility polygons [33] and have applied them to a number of problems in computational geometry, including the construction of a convex polygon nested between two given polygons and having the fewest possible vertices, and finding a polygonal path between two points that lies inside a polygon and has the fewest possible segments [34]. We have also developed an efficient method of computing the visibility graph of a set of polygons [35].

Other recent areas of investigation deal with finding packings and coverings of the plane by translates of a convex polygon [36] and decomposing a convex polygon into a vector sum of simpler convex polygons [37]. We

have also developed simple algorithms for recovering an orthogonal region (in space) from the set of its corner vertices, together with a specification of their numbers of incident edges and faces [38].

Digital geometry deals with geometric properties of subsets of digital images. A general introduction to digital geometry can be found in [39], and an abstract formulation in terms of "oriented graphs" is presented in [40]. Specific topics studied during the past year include metrics [41], straightness, and connectedness [42].

## 7. PARALLEL ALGORITHMS

Another ongoing research area is the study of parallel algorithms for geometric computations and other vision-related tasks, using various models of parallel computation. During the past year, work was done, in particular, on the following topics:

a) Mesh-connected computer algorithms for constructing and analyzing quadtree representations of images [43].

b) Hypercube algorithms for performing various basic data communication tasks, including parallel prefix [44] and permutation routing [45], which have applications (among others) to the manipulation of geometric data structures.

c) Shared-memory algorithms for processing medial axis (MAT) [46] and quadtree [47] image representations.

d) Bounds on the speedup of search algorithms (in particular, branch-and-bound) when implemented in parallel on shared-memory machines [48].

## 8. PYRAMID TECHNIQUES

We are continuing to investigate divide-and-conquer techniques for fast segmentation and analysis of images, suitable for implementation on "pyramid" or hypercube machines. An overview of this work can be found in [48]. Specific areas studied during the past year include histogramming [50], bimodality detection [51], texture-based image segmentation [52], curve detection [53] and description [54], and corner detection [55]. We are also studying the sensitivity of these algorithms to "noisiness" in the architecture. In this connection, we have developed methods of generating "stochastic" pyramids [56] and have applied them to the fast generation of hierarchies of random tessellations of an image [57].

## 9. OTHER TOPICS

During the past year we have also done work on a number of other topics related to image analysis and computer vision. This work is briefly outlined in the following paragraphs.

a) **Feature detection**

In the area of feature detection, we have developed a mask-matching approach to the local detection of curves and edges, have studied its behavior for noisy images [58], and have used consistent labeling methods to link the masks into features [59]. We have also developed a new set of template-matching edge operators using an associative mapping between ideal step edges and an orthonormal basis [60]. We have developed an efficient algorithm for computing two-dimensional Gaussian weights that requires only two multiplications per pixel [61].

On a more global level, we have investigated the importance of treating edges and curves two-dimensionally, rather than analyzing them as functions of a single parameter [62]. We have applied heuristic search techniques to the tracking of region boundaries in sets of serial section images [63]. We have developed nonparametric methods of fitting a straight line to noisy image data [64], and have more recently extended it to the fitting of multiple lines or planes [65].

b) **Matching**

We have investigated the problem of image matching from a theoretical point of view. We have studied the problem of computation of visual correspondence, given that we already know some values of the correspondence function. We have determined the mathematical constraints that enable us to grow a solution for the correspondence function from a point where its value is known, using the image intensity function. The results are applicable to many image matching problems, such as stereo image interpretation, object analysis, motion analysis, change detection and the like. [66]

We have also studied efficient algorithms for two-dimensional pattern matching in the presence of errors [67], in both sequential and parallel implementations [68].

We are also analyzing various types of matching tasks from a probabilistic standpoint. In particular, we have studied two-stage matching procedures as applied to labelled graphs and other domains relevant to computer vision. We do not require that the match be exact but only that it satisfy a specified error criterion. We have shown that it is computationally more efficient to initially match a subgraph and check the rest of the graph only when this match succeeds. A probabilistic analysis of the expected cost of this procedure has been given with the aim of determining the optimum subgraph size which minimizes this cost. The results were extended to graph matching with geometric constraints as well as to templates. [69]

Matching of data structures such as digital images or labeled graphs is computationally expensive, because it requires node-by-node comparisons of the labels. If we have probabilistic models for the classes of data structures being matched, we can reduce the expected computational cost of matching by comparing the nodes in an appropriate order. We have established some general results about this approach, and also obtained experimental results for digital images, when we know the probability densities of their gray levels, or more generally,

the probability densities of arrays of local property values derived from the images. **A paper on this work appears in these Proceedings.**

c) **Miscellaneous**

In the area of texture analysis, we have studied the problem of identifying a random field belonging to a given class, given a sample generated by that random field. We take the field to be from one of two special classes: stationary fields of independent samples and fields that are simple stationary Markov chains. Interval estimators for the parameters of the field have been derived from the joint frequencies of occurrence of elements of the sample. We used Monte Carlo simulations to evaluate the performance of these estimators and to investigate the tightness of some theoretical bounds for their confidence levels. [70]

In connection with image segmentation, we have developed algorithms based on minimization of compactness and of fuzziness whereby it is possible to obtain both fuzzy and nonfuzzy (thresholded) versions of an ill-defined image. The incorporation of fuzziness in the spatial domain, i.e., in describing the geometry of regions, makes it possible to provide more meaningful results than by considering fuzziness in grey level alone. [71]

We are also studying efficient models of neural networks and their application to the solution of optimization problems. Hopfield and Tank have shown that neural networks can be used in solving very complicated computational problems if they are formulated as optimization problems. Furthermore, they have shown that to obtain good solutions it is necessary to use analog networks rather than digital networks. Simulations of analog networks involve the solution of many coupled differential equations and therefore can be time consuming. For software implementation we have proposed a model of an analog network that does not involve differential equations, and thus is much more efficient. This is accomplished without compromising the quality of the solutions. Like Hopfield and Tank we have used the Traveling-Salesman Problem as an example. [72]

Finally, we have undertaken a general study of the effects of quantization errors on computer vision algorithms. Due to the important role that digitization error plays in the field of computer vision, a careful analysis of its impact on the computational approaches used in the field is necessary. We have developed the mathematical tools for the computation of the average error due to quantization. They can be used in estimating the actual error occurring in the implementation of a method. We have also derived the analytic expression for the probability density of error distribution of a function of an arbitrarily large number of independently quantized variables. The probability of the error of the function to be within a given range can thus be obtained accurately. In analyzing the applicability of an approach one must determine whether the approach is capable of withstanding the quantization error. If not, then regardless of the accuracy with which the experiments are carried out the approach will yield unacceptable results. The tools that we have developed can be used in the analysis of the applicability of a given algorithm, hence revealing the intrinsic limitations of the approach. [73] **A paper on this work appears in these Proceedings.**

## REFERENCES

The following references are technical reports from the Computer Vision Laboratory, Center for Automation Research, University of Maryland, College Park, MD 20742-3411.


[1] John (Yiannis) Aloimonos and Anargyros Papageorgiou, "On the Kinetic Depth Effect: Lower Bounds, Regularization and Learning." CAR-TR-261, CS-TR-1770, DAAB07-86-K-F073, January 1987.

[2] Minas E. Spetsakis and John (Yiannis) Aloimonos, "Closed Form Solution to the Structure from Motion Problem from Line Correspondences." CAR-TR-274, CS-TR-1798, DAAB07-86-K-F073, March 1987.

[3] Minas E. Spetsakis and John (Yiannis) Aloimonos, "Structure from Motion from Line Correspondences: New Results." CAR-TR-340, CS-TR-1966, DAAB07-86-K-F073, January 1988.

[4] Anup Basu and John (Yiannis) Aloimonos, "A Robust Algorithm for Determining the Translation of a Rigidly Moving Surface without Correspondence, for Robotics Applications." CAR-TR-279, CS-TR-1818, DAAB07-86-K-F073, March 1987.

[5] Tsai-Chia Chou and Ken-ichi Kanatani, "Recovering 3D Rigid Motion Without Correspondence." CAR-TR-297, CS-TR-1864, DAAB07-86-K-F073, June 1987.

[6] David Shulman and John (Yiannis) Aloimonos, "(Non)Rigid Motion Interpretation: A Regularized Approach." CAR-TR-295, CS-TR-1860, DAAB07-86-K-F073, June 1987.

[7] Eiki Ito and John (Yiannis) Aloimonos, "Determining Three Dimensional Transformation Parameters from Images: Theory." CAR-TR-318, CS-TR-1906, DAAB07-86-K-F073, August 1987.

[8] Randal C. Nelson and John (Yiannis) Aloimonos, "Finding Motion Parameters from Spherical Flow Fields (Or the Advantages of Having Eyes in the Back of your Head)." CAR-TR-287, CS-TR-1840, DAAB07-86-K-F073, April 1987.

[9] John (Yiannis) Aloimonos and Michael Swain, "Shape from Patterns: Regularization." CAR-TR-283, CS-TR-1826, DAAB07-86-K-F073, April 1987.

[10] John (Yiannis) Aloimonos, "Shape from Texture." CAR-TR-319, CS-TR-1907, DAAB07-86-K-F073, August 1987.

[11] John (Yiannis) Aloimonos and Michael Swain, "Paraperspective Projection: Between Orthography and Perspective." CAR-TR-320, CS-TR-1908, DAAB07-86-K-F073, August 1987.

[12] John (Yiannis) Aloimonos and Anup Basu, "Combining Information in Low-Level Vision." CAR-TR-336, CS-TR-1947, DAAB07-86-K-F073, November 1987.

[13] John (Yiannis) Aloimonos, "Combining Shading and Motion to Compute Shape and Light Source Direction." CAR-TR-262, CS-TR-1771, DAAB07-86-K-F073, October 1986.

[14] Tsai-Chia Chou, John (Yiannis) Aloimonos and Azriel Rosenfeld, "Correspondenceless Model Based and Active Perception of Shape from Contour." CAR-TR-275, CS-TR-1800, DAAB07-86-K-F073, March 1987.

[15] Minoru Asada, "Cylindrical Shape from Contour and Shading without Knowledge of Lighting Conditions or Surface Albedo." CAR-TR-276, CS-TR-1801, DAAB07-86-K-F073, March 1987.

[16] Steven Connelly, "Qualitative Approaches to Image Analysis using Color, Shading, and Highlights." CAR-TR-292, CS-TR-1854, DCR-86-03723, May 1987.

[17] Jacob Beck and Richard Ivry, "On the Role of Figural Organization in Perceptual Transparency." CAR-TR-347, CS-TR-1976, DAAB07-86-K-F073, January 1988.

[18] Rand Waltzman, "Finding Symmetries of Polyhedra." CAR-TR-333, CS-TR-1937, DAAB07-86-K-F073, October 1987.

[19] Isaac Weiss, "Projective Invariants of Shapes." CAR-TR-339, CS-TR-1965, DAAB07-86-K-F073, January 1988.

[20] Daniel DeMenthon, Tharakesh Siddalingaiah and Larry S. Davis, "Production of Dense Range Images with the CVL Light-Stripe Range Scanner." CAR-TR-337, CS-TR-1962, DACA76-84-C-0004, December 1987.

[21] Minoru Asada and Saburo Tsuji, "Shape from Projecting a Stripe Pattern." CAR-TR-263, CS-TR-1773, DACA76-84-C-0004, January 1987.

[22] Minoru Asada, "Building a 3-D World Model for a Mobile Robot from Sensory Data." CAR-TR-332, CS-TR-1936, DACA76-84-C-0004, October 1987.

[23] Sharat Chandran, Larry S. Davis, Daniel Dementhon, Sven J. Dickenson, Suresh Gajulapalli, Shie-Rei Huang, Todd R. Kushner, Jacqueline Le Moigne, Sunil Puri, Tharakesh Siddalingaiah and Phillip Veatch, "An Overview of Vision-Based Navigation for Autonomous Land Vehicles 1986." CAR-TR-285, CS-TR-1831, DACA76-84-C-0004, April 1987.

[24] Phillip A. Veatch and Larry S. Davis, "IRS: A Simulator for Autonomous Land Vehicle Navigation." CAR-TR-310, CS-TR-1889, DACA76-84-C-0004, July 1987.

[24] Sunil Puri and Larry S. Davis, "Two Dimensional Path Planning with Obstacles and Shadows." CAR-TR-255, CS-TR-1760, DACA76-84-C-0004, January 1987.

[26] Minoru Asada, "3-D Road Structure from Motion Stereo." CAR-TR-286, CS-TR-1839, DACA76-84-C-0004, April 1987.

[27] Sven J. Dickinson and Larry S. Davis, "An Expert Vision System for Autonomous Land Vehicle Road Following." CAR-TR-330, CS-TR-1932, DACA76-84-C-0004, October 1987.

[28] Phillip A. Veatch and Larry S. Davis, "Range Imagery Algorithms for the Detection of Obstacles by Autonomous Vehicles." CAR-TR-309, CS-TR-1888, DACA76-84-C-0004, July 1987.

[29] Randal C. Nelson and John (Yiannis) Aloimonos, "Using Flow Field Divergence for Obstacle Avoidance in Visual Navigation." CAR-TR-322, CS-TR-1914, DAAB07-86-K-F073, September 1987.

[30] Raju Prasannappa, Larry S. Davis and Vincent S.S. Hwang, "A Knowledge Based Vision System for Aerial Image Understanding." CAR-TR-253, CS-TR-1758, DMA800-85-C-0007, January 1987.

[31] Chengye Wang, Liuqing Huang and Azriel Rosenfeld, "Detecting Clouds and Cloud Shadows on Aerial Photographs." CAR-TR-268, CS-TR-1788, AFOSR-86-0092, February 1987.

[32] "Final Report on Contract DMA800-85-C-0007, Automated Feature Analysis." December 1985.

[33] Subir Kumar Ghosh, "Computing the Visibility Polygon from a Convex Set." CAR-TR-246, CS-TR-1749, AFOSR-86-0092, December 1986.

[34] Subir Kumar Ghosh, "A Few Applications of the Set-Visibility Algorithm." CAR-TR-273, CS-TR-1797, AFOSR-86-0092, March 1987.

[35] Subir Kumar Ghosh and David M. Mount, "An Output Sensitive Algorithm for Computing Visibility Graphs." CAR-TR-302, CS-TR-1874, AFOSR-86-0092, July 1987.

[36] David M. Mount and Ruth Silverman, "Packing and Covering the Plane with Translates of a Convex Polygon." CAR-TR-324, CS-TR-1917, AFOSR-86-0092, October 1987.

[37] Ruth Silverman and Alan H. Stein, "Algorithms for the Decomposition of Convex Polygons." CAR-TR-343, CS-TR-1969, DCR-86-03723, January 1988.

[38] Chung-Nim Lee and Azriel Rosenfeld, "Representation of Orthogonal Regions by Vertices." CAR-TR-257, CS-TR-1766, AFOSR-86-0092, January 1987.

[39] Robert A. Melter and Azriel Rosenfeld, "Digital Geometry." CAR-TR-323, CS-TR-1916, DCR-86-03723, September 1987.

[40] Reinhard Klette and Klaus Voss, "The Three Basic Formulae of Oriented Graphs." CAR-TR-305, CS-TR-1883, AFOSR-86-0092, July 1987.

[41] Robert A. Melter, "Some Characterizations of City Block Distance." CAR-TR-258, CS-TR-1767, DCR-86-03723, January 1987.

[42] Robert A. Melter and Azriel Rosenfeld, "New Views of Linearity and Connectedness in Digital Geometry." CAR-TR-345, CS-TR-1974, DCR-86-03723, January 1988.

[43] Yubin Hung and Azriel Rosenfeld, "Parallel Processing of Linear Quadtrees on a Mesh-Connected Computer." CAR-TR-278, CS-TR-1817, MMC-119, March 1987.

[44] Yubin Hung, "Hypercube Parallel Prefix Algorithms." CAR-TR-291, CS-TR-1853, MMC-119, May 1987.

[45] Yubin Hung, "Efficient Algorithms for Some Permutation Routings on Hypercubes." CAR-TR-312, CS-TR-1899, MMC-119, August 1987.

[46] Sharat Chandran, "Shared Memory Geometric Algorithms Based on the Medial Axis Transform." CAR-TR-284, CS-TR-1827, April 1987.

[47] Angela Y. Wu, S.K. Bhaskar and Azriel Rosenfeld, "Parallel Processing of Regions Represented by Linear Quadtrees." CAR-TR-296, CS-TR-1863, AFOSR-86-0092, June 1987.

[48] Shie-rei Huang and Larry S. Davis, "A Tight Upper Bound for the Speedup of Parallel Best-First Branch-and-Bound Algorithms." CAR-TR-290, CS-TR-1852, DACA76-84-C-0004, May 1987.

[49] Azriel Rosenfeld, "Pyramid Algorithms for Efficient Vision." CAR-TR-299, CS-TR-1866, DCR-86-03723, June 1987.

[50] Thor Bestul and Larry S. Davis, "On Computing Histograms of Images in $\log n$ Time Using Fat Pyramids." CAR-TR-271, CS-TR-1791, DACA76-84-C-0004, February 1987.

[51] Tsai-Yun Phillips, Azriel Rosenfeld and Allen C. Sher, "$O(\log n)$ Bimodality Analysis." CAR-TR-313, CS-TR-1900, DCR-86-03723, August 1987.

[52] Allen C. Sher and Azriel Rosenfeld, "Detecting and Extracting Compact Textured Objects Using Pyramids." CAR-TR-269, CS-TR-1789, DCR-86-03723, February 1987.

[53] Steven Connelly and Azriel Rosenfeld, "A Pyramid Algorithm for Fast Curve Extraction." CAR-TR-270, CS-TR-1790, DCR-86-03723, February 1987.

[54] Peter Meer, Sam Baugher and Azriel Rosenfeld, "Hierarchical Processing of Multiscale Planar Curves." CAR-TR-245, CS-TR-1748, DCR-86-03723, December 1986.

[55] Sam Baugher and Azriel Rosenfeld, "Corner detection and Localization in a Pyramid." CAR-TR-298, CS-TR-1865, DCR-86-03723, June 1987.

[56] Peter Meer, "Stochastic Image Pyramids." CAR-TR-301, CS-TR-1871, DCR-86-03723, June 1987.

[57] Peter Meer and Steven Connelly, "A Fast parallel Method for Synthesis of Random Patterns." CAR-TR-335, CS-TR-1945, DCR-86-03723, October 1987.

[58] Nathan S. Netanyahu and Azriel Rosenfeld, "Mask Matching for Linear Feature Detection." CAR-TR-254, CS-TR-1759, DMA800-85-C-0007, January 1987.

[59] J. John Kim and Azriel Rosenfeld, "Feature Detection Based on Pairwise Consistent Labeling." CAR-TR-272, CS-TR-1792, DMA800-85-C-0007, February 1987.

[60] Peter Meer, Shijie Wang and Harry Wechsler, "Edge Detection by Associative Mapping." CAR-TR-281, CS-TR-1822, DCR-86-03723, March 1987.

[61] Peter Meer, "Efficient Computation of Two-Dimensional Gaussian Windows." CAR-TR-329, CS-TR-1930, DCR-86-03723, October 1987.

[62] Steven Connelly and Azriel Rosenfeld, "On the Need for 2D Methods in the Analysis of Plane Curves." CAR-TR-277, CS-TR-1816, DCR-86-03723, March 1987.

[63] John Danilo Cappelletti, "Three-Dimensional Boundary Following." CAR-TR-314, CS-TR-1902, DCR-86-03723, August 1987.

[64] Behzad Kamgar-Parsi and Behrooz Kamgar-Parsi, "A Nonparametric Method for Fitting a Straight Line to a Noisy Image." CAR-TR-315, CS-TR-1903, DAAB07-86-K-F073, September 1987.

[65] Behrooz Kamgar-Parsi and Behzad Kamgar-Parsi, "Simultaneous Fitting of Several Planes to Point Sets Using Neural Networks." CAR-TR-346, CS-TR-1975, DAAB07-86-K-F073, January 1988.

[66] John (Yiannis) Aloimonos and Behrooz Kamgar-Parsi, "Correspondence from Correspondence." CAR-TR-260, CS-TR-1769, DAAB07-86-K-F073, January 1987.

[67] Kamala Krithivasan and R. Sitalakshmi, "Efficient Two Dimensional Pattern Matching in the Presence of Errors." CAR-TR-259, CS-TR-1768, AFOSR-86-0092, January 1987.

[68] Kamala Krithivasan, "Efficient Two Dimensional Parallel and Serial Approximate Pattern Matching." CAR-TR-266, CS-TR-1786, AFOSR-86-0092, February 1987.

[69] Ramesh Sitaraman and Azriel Rosenfeld, "Probabilistic Analysis of Two Stage Matching." CAR-TR-294, CS-TR-1858, DAAB07-86-K-F073, June 1987.

[70] Stanley M. Dunn, Richard L. Keizer and Azriel Rosenfeld, "Random Field Identification from a Sample: Experimental Results." CAR-TR-331, CS-TR-1934, October 1987.

[71] Sankar K. Pal and Azriel Rosenfeld, "Image Enhancement and Thresholding by Optimization of Fuzzy Compactness." CAR-TR-252, CS-TR-1757, DCR-86-03723, January 1987.

[72] Behzad Kamgar-Parsi and Behrooz Kamgar-Parsi, "An Efficient Model of Neural Networks for Optimization." CAR-TR-326, CS-TR-1922, DAAB07-86-K-F073, September 1987.

[73] Behrooz Kamgar-Parsi and Behzad Kamgar-Parsi, "Evaluation of Quantization Error in Computer Vision." CAR-TR-316, CS-TR-1904, DAAB07-86-K-F073, September 1987.

# CMU Image Understanding Program

**Takeo Kanade**

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

## Abstract

*The CMU Image Understanding Program covers a wide range of topics ranging from the basic computer vision issues in color, motion, and shape to the vision system issues in parallel architectures and outdoor navigation systems. This report reviews our progress since the December 1987 Image Understanding Workshop. Highlights of our progress in this period include:*

- *Continuous Stereo by Kalman Filtering*
- *Color Image Segmentation by Dichromatic Reflection Model*
- *Sensor Modeling for Model-Based Vision*
- *3D FORM System for Geometric Reasoning*
- *Navlab Demonstration Systems*
- *Range Data Analysis for Outdoor Scenes*
- *Precise Camera Calibration Process*
- *Microscopic Traffic Simulator*
- *Parallel Vision Software on Warp*
- *Scan Line Array Processors (SLAP)*

## 1. Physical and Computational Models for Low-Level Vision

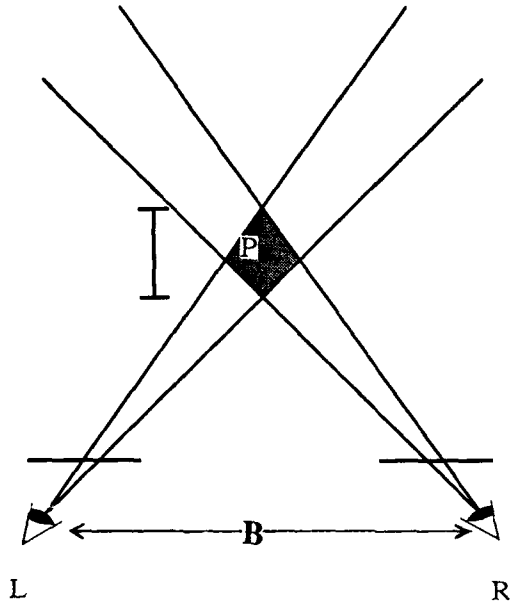
One of our successful approaches in low-level vision is to transform the physical and optical processes which underlie vision into computational models. This approach results in algorithms that are more powerful and revealing than traditional *adhoc* methods based solely on heuristic knowledge. The Calibrated Imaging Laboratory of CMU [Shafer 85a] has proven to provide critical support for the development and testing of those new physics- and optics-based algorithms. We have made substantial progress in the area of continuous motion stereo and color understanding.

### 1.1 Continuous Stereo by Kalman Filtering

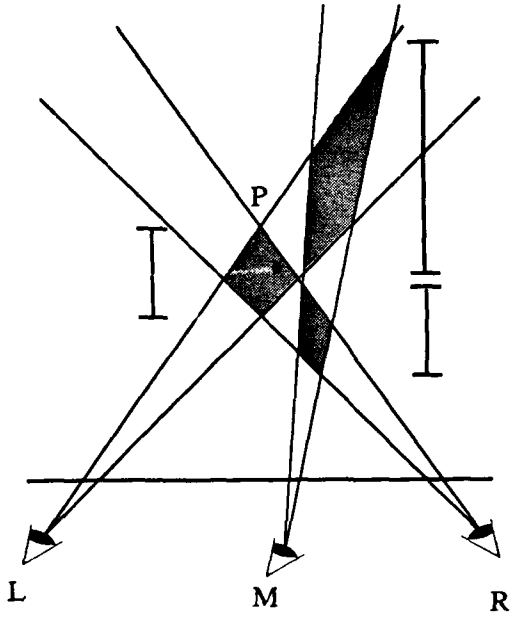
Using known camera motion to estimate depth from image sequences is an important problem in robot vision. Many applications of depth from motion, including navigation and manipulation, require algorithms that can estimate depth in an on-line, incremental fashion. This requires a representation that records the uncertainty, as well as the depth value itself, in depth estimates, and a mechanism that integrates new measurements with existing depth estimates to reduce the uncertainty over time. We (Matthies, Szeliski, and Kanade) have developed an algorithm that estimates depth and depth uncertainty, and incrementally refines these estimates over time.

The Matthies-Szeliski-Kanade on-line depth estimator is based on Kalman filtering. A correlation-based flow algorithm measures both the local displacement at each pixel or feature position and the confidence (or variance) of the displacement. These two "measurement images" are integrated with predicted depth and variance maps using a weighted least squares technique derived from the Kalman filter. Regularization-based smoothing [Szeliski 87] is used to reduce the noise in the flow estimates and to fill in areas of unknown disparity. The current maps of depth and uncertainty are extrapolated to the next frame by image warping, using the knowledge of the camera motion, and are resampled to keep the maps iconic. The algorithm processes an image sequence taken with small inter-frame displacements and produces an on-line estimate of depth that is refined over time. The algorithm can be either *iconic*, where the depth and uncertainty estimates are maintained at all the pixels, or *feature-based* where they are maintained at feature (ie., edge) points. Unlike most previous work [Broida , Chellapa 86] [Faugeras et al 86] [Matthies, Shafer 87] [Rives, Breuil, Espiau 86] which used Kalman filtering to keep track of a sparse set of trackable features, the new approach produces a dense map of depth and uncertainty estimations.

This work has revealed an interesting relationship which exists among the length of the baseline, the number of images, and the amount of uncertainty in stereo. In figure 1(a), assume we have a binocular stereo system with camera $L$ and camera $R$ separated by a baseline $B$. The dilemma in stereo is that as the baseline becomes longer, the depth measurement of the scene point $P$ becomes more accurate, but at the same time the matching (correspondence) becomes more difficult. Now, assume we place one more camera $M$ in the middle of the baseline as shown in figure 1(b). This creates two more stereo problems: one with cameras $L$ and $M$ and the other with




(a)




(b)

**Figure 1:** Stereo, baseline and uncertainty: (a) The wider the baseline ***B*** is, the more accurate is the depth measurement, but the more difficult is the matching; (b) The middle camera generates two more stereo matching problems, each of which is easier, but less accurate than the original one.

cameras ***M*** and ***R***. The matching problems for these intermediate two stereos must be easier than the original one with ***L*** and ***R***, even though the depth measurements by them would not be as good as that by the original stereo. If, however, the two new problems can be solved successfully, the original matching problem must have been be solved, too, because of the transitivity property. In fact, trinocular (three-eye) stereo systems that have been studied by several researchers exploit the additional constraints that the middle camera provides. The EPI analysis by Bolles and others [Bolles, Baker, Marimont 87] also exploits the geometrical constraints that the locus of ***P*** must satisfy in the image sequence obtained by moving camera ***M*** from position ***L*** to ***R***.

However, another interesting question here is whether the new middle camera ***M*** helps reduce the uncertainty (or increase the accuracy) in the depth measurement of ***P***. The answer is yes, and therefore putting the third camera helps to both simplify the original matching problem *and* reduce the uncertainty. Therefore we can add more camera positions between ***L***, ***M***, and ***R*** for further improvement. The analysis by Matthies, Szeliski, and Kanade proves that, if we assume an uncertainty $\sigma_e^2$ in the pixel positions, the uncertainty $\sigma^2(N)$ in the depth measurement by using $N$ cameras is,

$$\sigma^2(N) \sim \frac{\sigma_e^2}{N^3}.$$

and that this is $N$ times better (smaller) than the uncertainty obtained by the original stereo, This decrease of uncertainty at the rate of $N^3$ by using Matthies-Szeliski-Kanade on-line depth estimator has been experimentally verified: see [Matthies, Szeliski, Kanade 88]. An important advantage of this estimator is that it is incremental and does not require the processing of all $N$ images at once.

The above equation also has a practical significance. The decrease of uncertainty by using multiple images can be used for shortening the baseline, which reduces occlusion and matching problems. The experiments performed by Matthies, Szeliski and Kanade used ten images in which two consecutive images were taken with only 0.05 inches (1.27 mm) apart. Thus the total baseline was only 0.5 inches (1.27 cm), while the distance to the scene was about 20 to 40 inches (50 cm to 100 cm). The triangle for triangulation is a very sharp one; the ratio of the baseline to the height is approximately 1:80. Yet, they achieved accuracy up to 0.5 %.

## 1.2 Color Understanding

When we look at a color image, we can interpret it as a collection of shiny and matte surfaces, smooth and rough, interacting with light, shape, and shadow. However, computer vision has not yet been successful at deriving a similar description of surface and illumination properties from an image. The key reason for this failure has been a lack of models rich enough to relate pixels and pixel-aggregates to scene characteristics. In the past, with few exceptions, most work with color images has considered object color to be a constant property of an object, and color variation on an object was attributed to noise. However, color variation in real

scenes depends to a much larger degree on the optical reflection properties of the scene. This variation causes the perception of object color, highlights, shadows and shading scene properties that can be determined and used by color vision algorithms.

We (Klinker, Shafer, and Kanade) have taken an approach to color image understanding that accounts for color variations due to highlights and shading. The approach is based on Shafer's dichromatic reflection model which describes pixel colors as a linear combination of an object color and a highlight color [Shafer 85b]. All color pixels from one object then form a planar cluster in the color space. The cluster shape is determined by the object and highlight colors and by the object shape and illumination geometry. In addition to the color reflection model, we also use a sensor model which accounts for camera properties, such as a limited dynamic range, blooming, gamma-correction, and chromatic aberration. Such a model allowed us to obtain high quality color images (through color balancing and spectral linearization) in which most pixels maintain the linear properties of light reflection. As a result, our algorithms for color understanding are applicable to real images, instead of only to synthesized images.

In the 1987 Image Understanding workshop, Klinker, Shafer, and Kanade reported [Klinker, Shafer and Kanade 87] how the theory can be used to separate color images into two intrinsic images, one showing the scene without highlights, and the other one showing only the highlights. Previously, the method was applied to hand-segmented images. Since then, Klinker and others have developed an automatic segmentation method that is based on the extended dichromatic reflection model; for details see [Klinker, Shafer, Kanade 88] in these proceedings. The method alternates between generating hypotheses about the scene from the image data and verifying whether the hypotheses fit the image. The hypotheses relate object color, shading, highlights and camera limitations to the shapes of color clusters in local image areas. Using this control structure, driven by a physical model of light reflection, local and global properties of the scene, such as object and illumination colors, are incrementally identified, and individual pixels in the images are interpreted according to color and intensity changes at different places in the image. This method has successfully segmented images of multiple objects with different colors, which contain shading and highlights, into individual colored objects.

When compared with the traditional heuristic segmentation methods, this Klinker-Shafer-Kanade method is superior in a few important ways. The traditional color segmentation methods base their analysis on intensity or color differences or on a fixed set of user-defined features, such as intensity, hue and saturation. The new method can analyze color variations along arbitrary color axes and in arbitrary color planes. Traditional algorithms also cannot distinguish reliably between different edge types, such as highlight edges, material edges and shading or shadow edges, and they cannot account for camera limitations. In contrast, the Klinker-Shafer-Kanade method generates such physical information about the scene, which is used in the subsequent step to separate color images into two intrinsic reflection images: a body reflection image and a highlight image. The body reflection image can be used to generate hypotheses about object shapes and about the object materials [Healey, Binford 87]. The highlight image will also provide strong evidence for the position and color of the light source.

The hypothesis-based approach that Klinker, Shafer and Kanade took for image segmentation provides a new paradigm for low-level image understanding. The method gains its strength from using an intrinsic model of physical processes that occur in the scene. The results are intrinsic images and hypotheses which are closely related in their interpretation to the intrinsic model, being instantiations of concepts formulated in the model. The analysis alternates between a bottom-up step which generates hypotheses and a top-down step which applies the hypotheses to the images. The analysis thus consists of many small, complete interpretation cycles that combine bottom-up processing with feedback in top-down processing. This approach stands in contrast to traditional image segmentation methods which do not relate their analysis to intrinsic models and that also generally have a strictly bottom-up control structure. We feel that many low-level image understanding methods such as shape-from-x methods, stereo and motion analysis may be viewed and approached under this paradigm.

## 1.3 Regular Repetitive Texture

Texture is also a vital clue to surface and object properties for low-level vision. For understanding texture, we (Hamey) are studying the perception of regular texture repetitions. The central problem of regular repetitive textures in analysis is a chicken-and-egg problem: the texture element is difficult to define until the repetition has been detected, but at the same time the repetition cannot be found until the texture element is defined. The difficulty is compounded when the regular placement pattern is not strictly constant, but drifts in frequency and phase. Past work based on Fourier analysis and co-occurrence matrices had a severe limitation in the kind of repetitions that it could deal with.

Hamey has developed a method to analyze a range of repetitive patterns [Hamey 88]. His method is based on two central ideas. The first of these is the Dominant Feature Assumption, which states that a repetitive texture pattern should contain some particular feature within the texture element that is more visually prominent than all the others. By looking for repetitions of just this feature, it is possible to screen away the other data and arrive at a computationally tractable algorithm for texture analysis. The Dominant Feature Assumption does produce a limitation on our systems, but it is still very general. In particular, textures that are easy for people to perceive seem to correspond in general to textures that obey this assumption.

The second is a theory of repetition description that shows how any two-dimensional repetition can be characterized using two particular vectors in the image. The theory has led to a simple algorithm that detects the vectors to describe any

regular two-dimensional repetition. This algorithm uses a new definition of image connectedness based on the *six-connected neighborhood graph*, a graph that connects the nearest neighbor within each 60-degree sector around the feature points in the image. This graph has simple and important properties of computational geometry that make it easy to use for analyzing repetitive texture patterns. Furthermore, rather than imposing a single grid structure on the texture in an image region, the method allows the structure to vary systematically across the region. This allows it to be applied to the deformations that arise in real-world image texture such as patterns on fabric or perspective texture gradients (foreshortening) on a tall building. Figure 2 is an example of analysis results.

## 2. Model-Based Vision

We have been working on two topics in this area: automatic generation of objection recognition programs from models and development of a general framework for geometrical reasoning for vision.

### 2.1 Automatic Generation of Object Recognition Algorithms

Traditionally, a recognition program is generated by a human expert who examines the features of an object, develops a strategy for a recognition procedure, and writes a specialized program for the individual object. However, this "hand writing" of a recognition program requires a long time for programming and testing. We (Ikeuchi) have been working on precompilation techniques to automatically generate recognition programs from object models [Ikeuchi 87]. A bin-picking hand-eye system has been demonstrated at CMU; using photometric stereo as a major sensor, the generated recognition program of a part runs and locates parts in a pile and a PUMA arm picks it up and places it at a predefined location and orientation.

Automatic generation of a recognition program requires several key components:

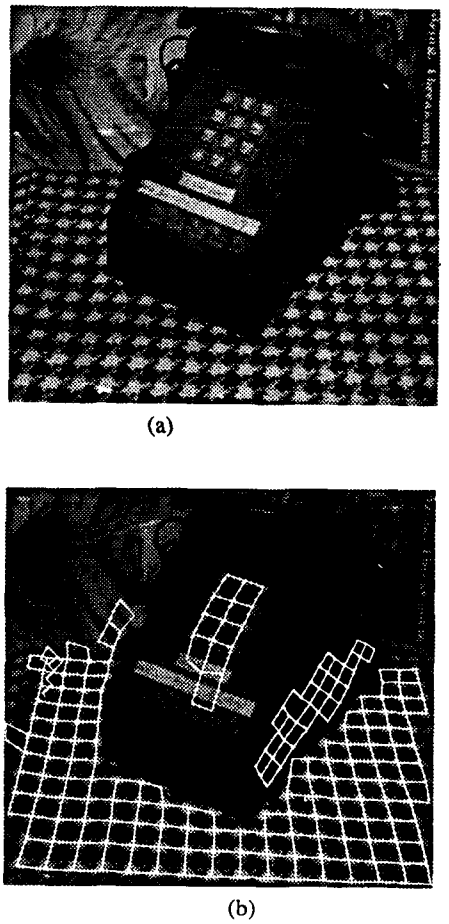
- *object models* to describe the geometric and photometric properties of an object to be recognized;
- *sensor models* to predict object appearances from the object model under a given sensor;
- *strategy generation* using the predicted appearances to produce a recognition strategy;
- *program generation* converting the recognition strategy to executable program.

During the last year, we have made a progress in sensor modeling and in program generation.

#### 2.1.1 Sensor Modeling

Different sensors, such as a video camera, light-stripe projector, or SAR (synthetic aperture radar), have very different properties and respond to different types of object features under different circumstances. Past work in computer vision has mostly been based on an assumption of a single type of sensor. This has created a built-in dependence on the sensor that makes existing vision programs restricted to the sensor for which they were designed. Our research in automatically generating algorithms for recognizing objects has revealed the necessity and the potential to break out of this restriction by using an explicit sensor model in addition to the explicit object model, allowing us to automatically generate appropriate algorithms to recognize an object from several different sensors.

Ikeuchi and Kanade [Ikeuchikanade 87, Ikeuchikanade 88] have developed a model for sensor properties that can specify two important characteristics: *detectability* and *reliability*. Detectability refers to the kind of features that can be detected, such as faces, edges, and vertices. For example, an edge detector is sensitive to edges; a laser range scanner is sensitive to faces; SAR is most sensitive to concave corners. We have developed a uniform representation for such



(a)

(b)

**Figure 2:** Result of analysis of regular repetitive textures. (a) input image; (b) Extracted grid patterns are overlaid. The grid on the side of the telephone is due to refletion of the textile pattern on the table.

detectability properties that allows many different sensor modalities to be described in a single framework. Reliability specifies a confidence value for the detection process, as well as a measure of how errors are propagated from the measured data to the inferred geometric features.

We have used this sensor modeling methodology to construct a survey of commonly available sensors, and produced detailed descriptions of photometric stereo and light-stripe range finders as examples. We plan to use these sensor models in conjunction with our previous methodology for generating recognition programs from solid models of objects, in order to produce a system that can operate with diverse sensors. This capability will be important for sensor fusion or integration tasks that involve the use of many sensors to recognize a single object, and will also be important in robot system design as an automated aid to sensor selection for specific applications. While doing this research, we came to realize the limitations of currently available geometric modeling systems, and have developed a frame-based geometrical modeling system VANTAGE which has a completely open architecture [Kanadeetal 88].

### 2.1.2 Object-Oriented Programming

A recognition strategy for an object specifies the kinds and the order of features to be used for classifying an input image into one of several representative appearance groups (called aspects). In our method, the recognition strategy is given as an interpretation tree in which each node represents a step of feature matching and classification.

Chang has been working on a systematic method to convert this strategy into an executable program which can actually access the images, compute the features, and make decisions [Chang 87]. Their method is based on an object-oriented programming technique. An object in object-oriented programming is a processing unit, which can store several internal values in slots. We can define demon functions for each slot, where a demon function will be invoked implicitly whenever we retrieve a value from the slot or insert a value into the slot. In our implementation, we use modified Framekit+, developed at CMU [Carbonell, Joseph 86].

Two kinds of objects have been prepared: *data objects* and *event objects*. A data object is used for representing geometric objects (such as edge and region) and extracting features from geometric objects. An event object is used for feature matching and attitude determination. A library of prototypical objects is prepared and the executable program is constructed by properly selecting and instantiating modules from it. The object-oriented programming paradigm provides modularity and extensibility.

This method has been applied to the generation of a recognition program for a toy wagon. The toy wagon has a fairly complex shape with seventeen attitudes. The resultant recognition program contains two kinds of data objects and six kinds of event objects. The generated program has been tested with real scenes and has recognized the wagon in a pile.

## 2.2 3D FORM: A Framework for Geometrical Reasoning

Three-dimensional object description and geometrical reasoning is critical for many applications of image understanding, because geometric relationships among objects are a rich source of knowledge and constraints for image analysis. Unfortunately, most 3D image understanding systems have utilized very limited solid or surface models and limited reasoning capabilities, and therefore cannot take advantage of the specific properties and relationships of any given image Our research in this area is aimed at developing a more general framework for representing 3D models and relationships, so that vision systems can use the specific information contained in each image to its best advantage.

We (Walker) have been developing a geometric reasoning system called 3DFORM [Walker, Herman, Kanade 87], using the frame language Framekit defined on CommonLISP. 3D FORM system includes a number of features that make it an improvement over past systems:

- 3DFORM uses frames to model object parts and geometric relations, which allows the system to be extended easily to incorporate new features.
- It is possible to specify shapes incompletely or by constraints on them, rather than direct complete description.
- 3DFORM includes explicit modeling of the projections from the 3D scene to the 2D image and back, which allows a program to reason back-and-forth as needed.
- Active procedures can be attached to the frames to dynamically compute values as needed, avoiding unnecessary computations.
- The order of computation is controlled by accessing objects' attribute values, which allows the system to perform top-down and bottom-up reasoning as needed.
- There is no need for an external "focus of attention" mechanism, which in past systems has sometimes been a complex and problematic item to construct.

During the last year, we have been extending the capabilities of 3DFORM to model more complex parts and relationships, such as the relationship of polygon vertices to the lines that form the edges of the polygon. When these relationships are evaluated, the side-effects include creating hypotheses for the missing or incomplete parts of each object. We have also been adding a mechanism for relationships at different levels of the part-whole hierarchy to interact with each other. Then when one feature is added to the interpretation, it can trigger reasoning processes at other levels of the hierarchy. This provides a rich structure for combining top-down and bottom-up reasoning in a single mechanism. Our test tasks for 3DFORM include completing a description of flat-roofed buildings from partial specifications, and reasoning about an object's shape from shadow and light source relationships.

## 3. Vision and System for Navigation

In the Strategic Computing Vision program, we (Thorpe et.al) have been working on developing vision techniques and integrating them into a demonstrable vehicle [Kanade, Thorpe, Whittaker 86, Thorpe, Hebert, Kanade, Shafer 87, Goto, Stentz 87]. During 1987, we have worked on several component technologies including range data analysis, object (car) recognition, calibration, map revision, and path planning. In addition, we have demonstrated two integrated systems, one in May and another in December, each of which integrated new advanced features of component technologies. Work was also begun on a simulator for research on driving on public streets.

### 3.1 Navlab and Demonstrations of Integrated Systems

The Navlab, short for Navigation Laboratory, is CMU's mobile robot for integration research. It is a self-contained laboratory, based on a commercial van chassis, with hydraulic drive and electric steering. Computers can steer and drive the van by electric and hydraulic servos, or a human driver can take over control if necessary. It is even licensed in Pennsylvania, so we can drive it by hand to our remote test sites. The feature of the Navlab that distinguishes it from other mobile robot research testbeds is that it can carry computers, sensors, and researchers onboard. Current major onboard equipment includes four SUN 3 workstations, a 100-MFLOP Warp systolic array computer, two color cameras, and an ERIM scanning laser range finder.

In May and December 1987, we demonstrated two integrated systems. The May 87 system used exactly the same algorithms for road following with obstacle detection as the November 86 system, but the color processing component was converted to Warp programs to increase the navigation speed. We have achieved the speed of 50 to 100 cm/sec which is five to ten times faster than the November 86 system.

The December 87 system had a much more ambitious goal and integrated several new capabilities. The new features of the December 87 system include:

- The system is based on the second version of CODGER [Shafer, Stentz, Thorpe 86, Stentz 88].
- The road following program with color analysis recognizes intersections explicitly.
- The range analysis compiles a terrain map from a sequence of ERIM range images.
- The path planner also uses uncertainty of road positions and vehicle controllability in planning a path [Stentz 88].
- The system starts with a topologically correct, but rough sketch of the road map, and as it navigates, the map reviser revises the map to include more precise geometry of the road and location of objects (mostly tress).
- The Warp system is divided into two 5-cell systems, which processes color images and range images individually.

This system was successfully demonstrated at the beginning of December 1987 at a very slow speed of 10 to 15 cm/sec. Figure 3 shows the initial map and the results of map revision at different detail levels.

### 3.2 Range Data Analysis for Outdoor Navigation

The analysis of range data is an important sensory component for an autonomous vehicle which navigates in an environment with many three-dimensional features such as trees, uneven terrain, and man-made objects. The Navlab is equipped with a scanning laser range finder which provides a range image of 256x64 pixels in half a second, together with an intrinsic reflectance image. We have been developing a variety of techniques of range data analysis for obstacle detection, surface description, terrain map building, and object recognition [Hebert, Kanade 86]. Figure 4 shows the flow of analysis. The paper [Hebert, Kanade 88] in these proceedings describes the recent development in detail.

Obstacle detection is a minimal capability required by an autonomous vehicle in order to navigate safely. This is done by first transforming the raw ERIM image into a bucket representation of an elevation map, and then identifying positions at which the elevation exhibits a large discontinuity and points at which the surface slope is above a given threshold.

For navigating on uneven terrain or recognizing 3D objects, a more accurate description of the environment is required. We describe surfaces by a set of connected surface patches. Each patch corresponds to a smooth portion of the surface and is approximated by a parameterized surface. In addition to the parameters and the neighbors, each region has two uncertainty factors: $\sigma_a$ and $\sigma_d$. $\sigma_a$ is the variance of the angle between the measured surface normal and the surface normal of the approximating surface at each point. $\sigma_d$ is the variance of the distance between the measured points and the approximating surface.

We build a terrain map building by integrating surface descriptions from different vantage points into a consistent 3D map. We have included the map building techniques in the Carnegie-Mellon NAVLAB system. A terrain map was maintained over a hundred meters while the vehicle was running autonomously under control of the road following program. Registration of terrain descriptions between frames is done by matching. The current vehicle estimate was used as an initial estimate for the matching. The features used for the matching part are the primitives of the surface description, the location of discrete objects, and location of the road edges extracted from the reflectance channel. The features are weighted according to their uncertainty; for example, the variance $\sigma_a$ is the weight in the case of planar features.

### 3.3 Object Recognition: Range and Color

Car recognition was selected as a sample problem of three-dimensional object recognition in the context of outdoor navigation. We have written two programs for that task: one uses range data as input and the other uses a color image. The car recognition program by range data searches for a

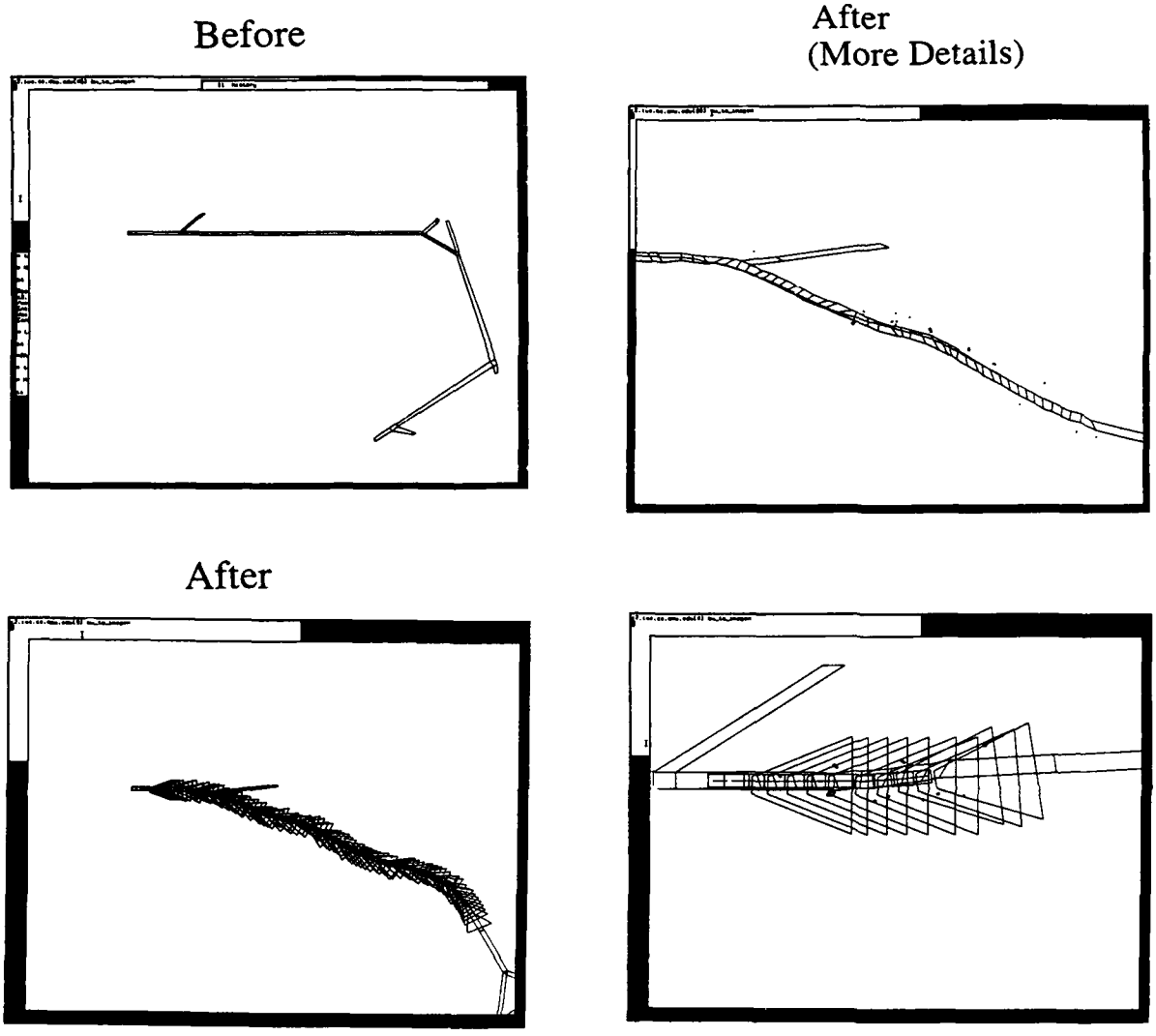



**Figure 3:** Map revision. The initial schematic map is only topologically correct. The revised map includes a detailed shape of the road as well as detected 3D objects (most of them trees and others are people.)

combination of surface patches $(S_{j1},..,S_{jn})$ from the surface description of the input scene which matches with the model. The model has two components: a set of surface primitives $(M_{i1},...,M_{in})$ and a set of constraints $C_k$. The constraints encapsulate knowledge about the object's shape, such as "surfaces $M_i$ and $M_j$ are orthogonal". A constraint, $c$, associated with a set of regions $(M_{i1},...,M_{in})$ can be viewed as a function that evaluates whether a partial matching $((M_{i1},S_{j1}),...,(M_{in},S_{jn}))$ is acceptable. A rule-based matcher has been developed and tested with range images which may contain cars in various orientations. [Hebert, Kanade 88] described the details.

A color image-based car recognition program has been also written. This program starts by detecting color edges. Lines and ellipses are extracted. The program searches for a combination of two or three trapezoids and ellipses which satisfy certain geometrical constraints so that they could be roof, windows, or car wheels. Once such a combination is found, hypotheses about possible car orientations are generated. Other features to support or reject the hypotheses are searched in top-down manner. Figure 5 shows several examples of the results from this program. We plan to combine these two programs, range-based and color-based, so that the recognition is reliable and fast.

### 3.4 Camera Calibration

Geometric camera calibration is the process of determining a mapping between points in world coordinates and the corresponding image locations of the points. Calibration must provide answers to both the *projection* problem (i.e., given the location of a point in space, predict its location in the image) and the *back-projection* problem (i.e., given a pixel in the image, compute the line-of-sight vector through the pixel). In previous methods, calibration typically involved the iterative solution to a system of non-linear equations, and some methods provide only a solution to one of the above two problems. We (Gremban, Thorpe and Kanade) have developed a method for performing camera calibration that provides a complete, accurate solution, using only linear systems of equations [Gremban, Thorpe, Kanade 88].

The method is based on the two-plane calibration method [Martins, Birk, Kelley 81]. By using two calibration planes, a line-of-sight vector is defined for each pixel in the image. The effective focal point of a camera can be obtained by solving the system that defines the intersection point of the line-of-sight vectors. Once the focal point has been determined, a complete camera model can be obtained with a straightforward least squares procedure. This method of geometric camera calibration has the advantages of being accurate, efficient, and practical for a wide variety of applications. The tests of the method were conducted using the Calibrated Imaging Laboratory (CIL) at CMU [Shafer 85a]. Positions of calibration points in the CIL were measured by the use of theodolites (surveyor's transits) Objects to be measured are placed at one end of an optical bench; the theodolites are fixed to the other end, separated by a little more than 1 meter.

The test results show that the accuracy achievable with a standard commercial CCD (Sony AVC-D1) using the standard 16mm lens supplied with the camera is as good as 1 part in 1400 (0.4 mm over 530 mm) to 1 part in 3500 (0.15 mm over 530 mm) depending the interpolation functions used. We plan to use this calibration method for calibrating and registering three cameras and a laser range finder mounted on the CMU Navlab.

### 3.5 Traffic Simulation

A microscopic traffic simulator called PHAROS was developed to study robot driving on public streets [Reece, Shafer 87]. PHAROS encodes detailed information about the topology of the street network, the geometry of the streets, the nature of all surface markings, the locations of signs, and the indications of traffic signals. In addition, PHAROS simulates a fleet of vehicles moving realistically through the network; these vehicles accelerate, brake, negotiate intersections, and change lanes according to a simple driver model. The simulated network and vehicles are displayed in near real-time with animated graphics so that vehicle behavior can be observed directly.

The appearance of the streets and the positions of the vehicles is used to compute the perceptual input that a robot driver would receive. A driving program will be developed to interpret the perceptual input and compute the appropriate robot actions. PHAROS will then move the robot in simulation. The simulator will allow us to study planning architectures and driving algorithms in parallel with our work in basic vision.

## 4. Parallel Architectures and Algorithms for Vision

We continue to develop tools and application software for parallel vision on Warp. In addition, we have been cooperating with VLSI group for their development of a new parallel vision architecture called SLAP.

### 4.1 Parallel Vision on Warp

Warp is the Carnegie Mellon Systolic Array Machine providing 100 MFLOP. Warp was designed by the Warp group at CMU and production (PC) Warp machines are manufactured by GE. As part of Strategic Computing Vision, we (Webb et. al) have been developing vision software for use by vision researchers [Annaratone, et al. 87]. As the software environment improves, Warp has begun to be used extensively for both Navlab computation and everyday vision research activities.

In 1987, Warp was installed on Navlab, and was used as a critical component in both the May 87 and December 87 systems. Programs which run on Warp include color image processing for road following and ERIM range data analysis for obstacle detection and surface description. In addition to outdoor navigation problems, Warp has been used for NMR image processing and adaptive beam forming for sonar by singular value decomposition.

WEB library is a collection of low to intermediate level vision routines which run on Warp. It currently contains more than 130 routines. The average speed-up obtained is a factor of 250 relative to the speed of a VAX 11/780. About half of WEB library has working validation procedures that record the actual execution times, so that they can be used to monitor the change made to the Warp system both in software and hardware.

The main high-level programming language for vision on Warp is *Apply*. One of the most important obstacles standing in the way of widespread use of parallel computers for low-level vision is the lack of a programming language that can be mapped efficiently onto different computer architectures which is suited for low-level vision. Apply [Hamey, Webb, Wu 87] was developed to generate efficient programs for a variety of parallel machines given a single source code. By simply describing the local operations on a local window, the Apply compiler can generate codes for the Warp machine (in W2) which execute the operations on the whole image efficiently. In fact, 80% of routines in WEB library are written in Apply. Currently Apply can generate code for not only the Warp machine, but also the Hughes Aircraft Corporation Hierarchical Bus Architecture and the Sun 3 workstation. A paper by Wallace, Hamey and Webb in these proceedings [Wallace, Webb, Wu 88] describes a recent

progress in Apply.

Because of the machine independence of the Apply language, programs written in Apply can be ported from one machine to another simply by recompilation. Moreover, the Apply compiler and the WEB library allow the comparison of the performance of vision machines, since the same source code will be running on both machine. This is the strongest possible basis for comparison of two computers.

Apply addresses the needs of the vision community for a simple, machine-independent, language for programming low-level vision algorithms. We are interested it seeing it mapped onto other architectures, and, in fact, a number of groups have already expressed interest. We are exchanging software with these groups to help them in developing Apply and WEB on their systems.

The next step is to develop a machine-independent language for other parts of computer vision. This language should be able to perform operations in low, mid, and feature-level vision, should be simple to program in, and potentially be mapped to a many diverse computer architectures. We are developing a specification for this language, which will be called *Adapt*, and are beginning implementation efforts.

## 4.2 Scan Line Array Processors, SLAP

Fisher, Highnam, and Rockoff of the VLSI group have been developing a Scan Line Array Processor (SLAP) for computer vision. It is a special-purpose VLSI processor array which includes one processing element (PE) for each pixel on a single image scan line, arranged in a linear array. Each PE has internal fixed-point functional units and a register bank. A SLAP operates in SIMD fashion, with a system controller broadcasting a single instruction to all PEs in each cycle. Adjacent processing elements can exchange data in each cycle, and a dedicated video shift register performs concurrent image I/O.

The simple control structure and linear topology of a SLAP lend themselves to cheap, fast implementations and to easy scaling with technology improvements. Given this promise of great cost effectiveness, a key issue is the efficiency with which problems can be mapped onto the array. We have designed a number of algorithms and algorithm mapping schemes that demonstrate efficient mappings for a broad range of important image operations, both local and global.

In order to fully assess the practical value of the SLAP approach, we are constructing a prototype array with 512 processing elements. Built around custom 2 micron CMOS

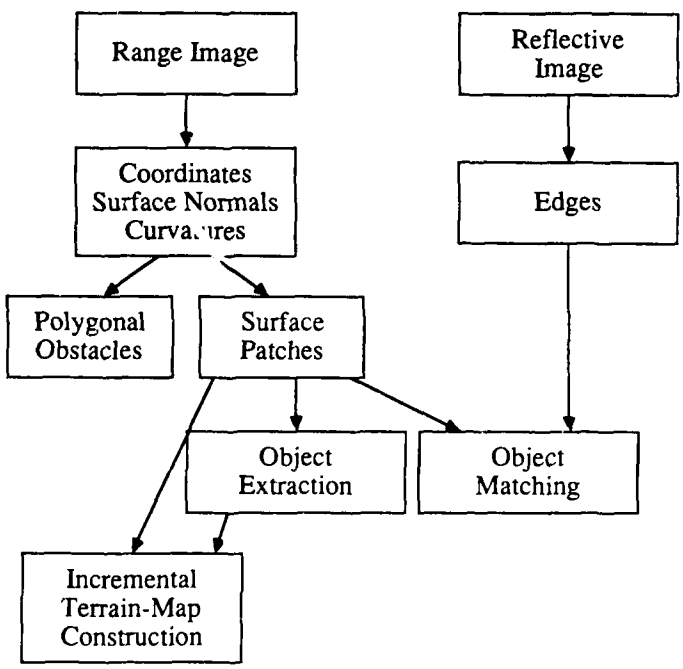



**Figure 4:** Flow of range data analysis

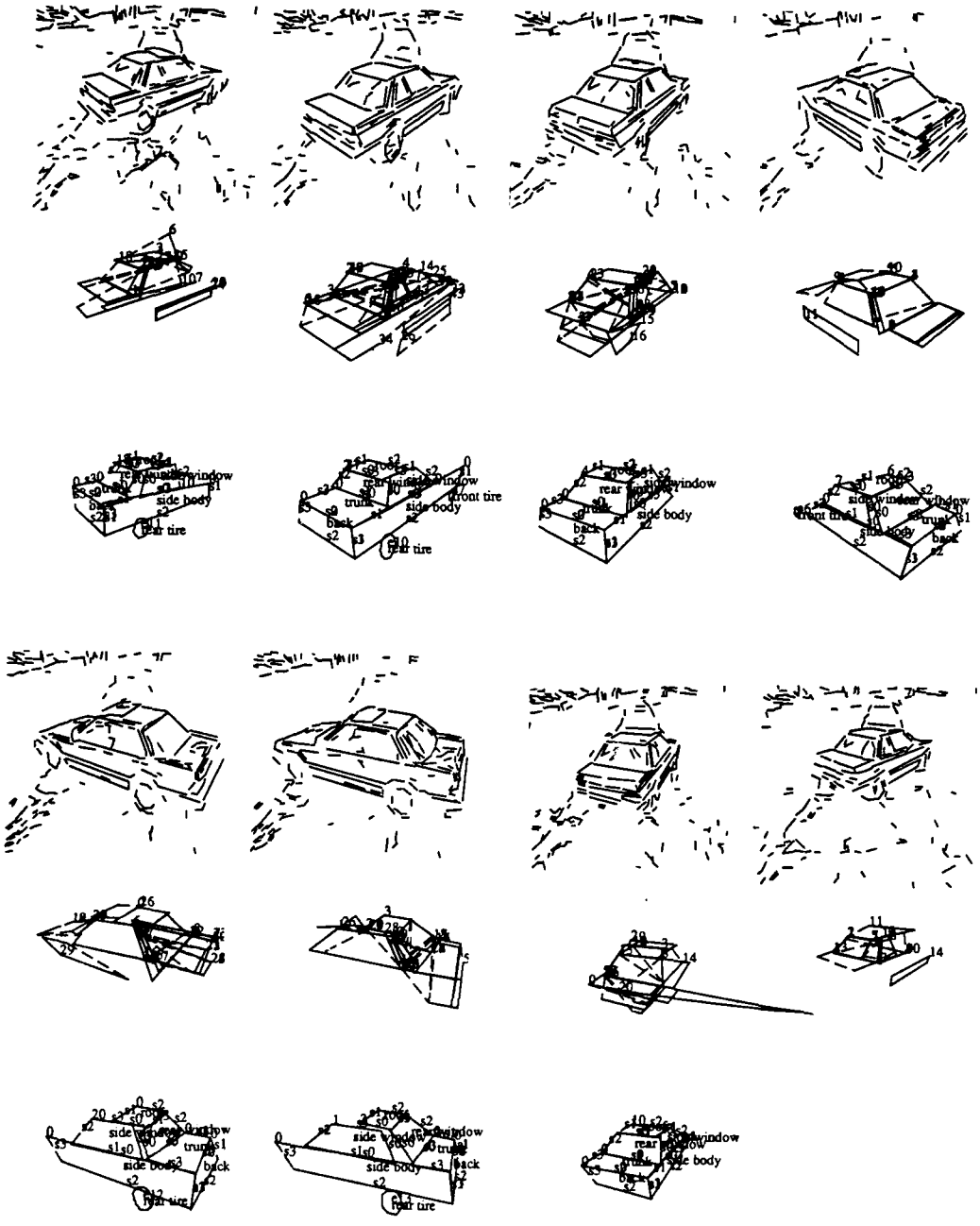



**Figure 5:** Results of car recognition by color images: The first row shows input edge images, and the last row shows the final results.

chips, the array and its controller will occupy three standard-size circuit cards and deliver some 4 billion 16-bit integer operations per second. Performance estimates on some commonly used algorithms indicate that this raw throughput can be put to good use. In addition to developing hardware and algorithms, we are also building low-level programming tools and an optimizing compiler for a high-level parallel language tailored for image processing. A paper by Fisher, Highnam, and Rockoff [Fisher, Highnam, Rockoff 88] in these proceedings gives an overview of the SLAP concept and the current implementation work, and provides some comparative performance predictions.

## References


[Annaratone, et al. 87]
M. Annaratone, F. Bitz, J. Deutch, L. Hamey, H. T. Kung, P. Maulik, H. Ribas, P. Tseng, and J. Webb.
Applications Experience on Warp.
In *Proceedings of the 1987 National Computer Conference*. AFIPS, 1987.

[Bolles, Baker, Marimont 87]
R. C. Bolles, H. H. Baker, and D. H. Marimont.
Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion.
*International Journal of Computer Vision* 1:7-55, 1987.

[Broida , Chellapa 86]
T. J. Broida and R. Chellappa.
Kinematics and Structure of a Rigid Object from a Sequence of Noisy Images.
In *Proc. Workshop on Motion: Representation and Analysis*, pages 95-100. IEEE, May, 1986.

[Carbonell, Joseph 86]
J. Carbonell and R. Joseph,
*Framekit+:A knowledge representation system.*
Technical Report CS-TR, Computer Science Department, Carnegie Mellon University, March, 1986.

[Chang 87]
H. Chang.
*A vision algorithm generator by object-oriented programming.*
Technical Report, Dept. of Electrical and Computer Engineering, Carnegie Mellon University, July, 1987.

[Faugeras et al 86]O. D. Faugeras, N. Ayache, B. Faverjon, and F. Lustman.
Building Visual Maps by Combining Noisy Stereo Measurements.
In *IEEE International Conference on Robotics and Automation*, pages 1433-1438. IEEE, San Francisco, California, April, 1986.

[Fisher, Highnam, Rockoff 88]
A. L. Fisher, P. T. Highnam, and T. E. Rockoff.
Scan Line Array Processors: Work in Progress.
In *Proc. of Image Understanding Workshop 1988*, pages (in these proceedings). Morgan Kaufmann Publishers, Inc., Cambridge, Massachusetts, April, 1988.

[Goto, Stentz 87] Y. Goto and A. Stentz.
Mobile Robot Navigation: The CMU System.
*IEEE Expert* :44-54, Winter, 1987.

[Gremban, Thorpe, Kanade 88]
K. D. Gremban, C. E. Thorpe, and T. Kanade.
Geometric Camera Calibration Using Systems of Linear Equations.
In *Proc. of Image Understanding Workshop 1988*, pages (in these proceedings). Morgan Kaufmann Publishers, Inc., Cambridge, Massachusetts, April, 1988.

[Hamey 88] Leonard Hamey.
*Computer Perception of Repetitive Textures.*
PhD thesis, Carnegie Mellon University, February, 1988.

[Hamey, Webb, Wu 87]
L. G. C. Hamey, J. A. Webb, and I.-C. Wu.
*Low-Level Vision on Warp and the Apply Programming Model.*
Technical Report CMU-RI-TR-87-17, Carnegie Mellon University, The Robotics Institute, July, 1987.

[Healey, Binford 87]
G. Healey and T.O. Binford.
Local Shape from Specularity.
In *Proceedings of the First International Conference on Computer Vision (ICCV)*, pages 151-161. IEEE, London, June, 1987.

[Hebert, Kanade 86]
M. Hebert, and Kanade, T.
Outdoor Scene Analysis Using Range Data.
In *IEEE International Conference on Robotics and Automation*. 1986.

[Hebert, Kanade 88]
M. Hebert and T. Kanade.
3-D Vision for Outdoor Navigation by an Autonomous Vehicle.
In *Proc. of Image Understanding Workshop 1988*, pages (in these proceedings). Morgan Kaufmann Publishers, Inc., Cambridge, Massachusetts, April, 1988.

[Ikeuchi 87]
K. Ikeuchi.
Generating an Interpretation Tree from a CAD Model for 3-D Object Recognition in Bin-Picking Tasks.
*International Journal of Computer Vision* 1(2):145-165, 1987.

[Ikeuchikanade 87]
K. Ikeuchi and T. Kanade.
*Modeling sensor detectability and reliability in the sensor configuration space.*
Technical Report CMU-CS-87-144, Carnegie-Mellon University, Computer Science Department, 1987.

[Ikeuchikanade 88]
K. Ikeuchi and T. Kanade.
Modeling Sensors and Applying Sensor Model to Automatic Generation of Object Recognition Program.
In *Proc. of Image Understanding Workshop 1988*, pages (in these proceedings). Morgan Kaufmann Publishers, Inc., Cambridge, Massachusetts, April, 1988.

[Kanade, Thorpe, Whittaker 86]
T. Kanade, C. Thorpe, and W. Whittaker.
Autonomous Land Vehicle Project at CMU.
In *Proc. ACM Computer Conference*. Feb, 1986.

[Kanadeetal 88] T. Kanade, P. Balakumar, J. C. Robert, R. Hoffman, and K. Ikeuchi.
Overview of geometric/sensor modeler VANTAGE.
In *Proc. The International Symposuim of Exposition on Robots*. The Australian Robot Association, Sydney, Australia, November, 1988.

[Klinker, Shafer and Kanade 87]
G. Klinker, S. A. Shafer, and T. Kanade.
Using a Color Reflection Model to Separate Highlights from Object Color.
In *Proc. of DARPA Image Understanding Workshop 1987*, pages 614-619. 1987.
Also to appear in *International Journal of Computer Vision (IJCV.*

[Klinker, Shafer, Kanade 88]
G. J. Klinker, S. A. Shafer, and T. Kanade.
Image Segmentation and Reflection Analysis Through Color.
In *Proc. of Image Understanding Workshop 1988*, pages (in these proceedings). Morgan Kaufmann Publishers, Inc., Cambridge, Massachusetts, April, 1988.

[Martins, Birk, Kelley 81]
H. A. Martins, J. R. Birk, and R. B. Kelley.
Camera Models Based on Data from Two Calibration Planes.
*Computer Graphics and Image Processing* 17:173-180, 1981.

[Matthies, Shafer 87]
Larry Matthies and Steven A. Shafer.
Error Modeling in Stereo Navigation.
*IEEE Journal on Robotics and Automation* :239-248, June, 1987.

[Matthies, Szeliski, Kanade 88]
L. Mathies, R. Szeliski, and T. Kanade.
Kalman Filter-Based Algorithms for Estimating Depth from Image Sequences.
In *Proc. of Image Understanding Workshop 1988*, pages (in these proceedings). Morgan Kaufmann Publishers, Inc., Cambridge, Massachusetts, April, 1988.

[Reece, Shafer 87]D. Reece and S. Shafer.
An Overview of the PHAROS Traffice Simulator.
In *Proc. of the 2nd International Conference on Road Safety.* x, Groningen, The Netherlands, September, 1987.

[Rives, Breuil, Espiau 86]
P. Rives, E. Breuil, and B. Espiau.
Recursive Estimation of 3D Features Using Optical Flow and Camera Motion.
In *Proceedings of Conference on Intelligent Autonomous Systems*, pages 522-532. Elsevier Science Publishers, Dec., 1986.

[Shafer 85a] S. A. Shafer.
The Calibrated Imaging Lab Under Construction at CMU.
In *Proc DARPA Image Understanding Workshop*, pages 519-515. 1985.

[Shafer 85b] S. A. Shafer.
Using Color to Separate Reflection Components.
*Color Research and Application* 10(4 (Winter)):210-218, 1985.

[Shafer, Stentz, Thorpe 86]
S. Shafer, A. Stentz, C. Thorpe.
An Architecture for Sensor Fusion in a Mobile Robot.
In *IEEE International Conference on Robotics and Automation.* 1986.

[Stentz 88] A. Stentz.
*The NAVLAB System for Mobile Robot Navigation.*
PhD thesis, Carnegie Mellon University, 1988.
(in preparation).

[Szeliski 87] Richard Szeliski.
Regularization uses Fractal Priors.
In *Proceedings of AAAI*, pages 749-754. Seattle, Washington, June, 1987.

[Thorpe, Hebert, Kanade, Shafer 87]
Charles Thorpe, Martial Hebert, Takeo Kanade, and Steven Shafer.
Vision and Navigation for Carnegie Mellon Navlab.
*Annual Review of Computer Science.*
Annual Reviews Inc., California, 1987, pages 521-556.

[Walker, Herman, Kanade 87]
Ellen Lowenfeld Walker, Martin Herman, and Takeo Kanade.
A Framework for Representing and Reasoning about Three-Dimensional Objects for Vision.
In *Proceedings of the AAAI Workshop on Spatial Reasoning and Multisensor Fusion*, pages 21-33. Morgan Kaufmann Publishers, Inc., October, 1987.

[Wallace, Webb, Wu 88]
R. S. Wallace, J. A. Webb, and I.-C. Wu.
Machine-Independent Image Processing: Performance of Apply on Diverse Architectures.
In *Proc. of Image Understanding Workshop 1988*, pages (in these proceedings). Morgan Kaufmann Publishers, Inc., Cambridge, Massachusetts, April, 1988.

# IMAGE UNDERSTANDING RESEARCH AT SRI INTERNATIONAL

Martin A. Fischler and Robert C. Bolles

Artificial Intelligence Center
SRI International
333 Ravenswood Avenue, Menlo Park, California 94025

February 5, 1988

## Abstract

The Image Understanding research program at SRI International is a broad effort spanning the entire range of machine vision research. In this report we describe the progress in two programs: the first is concerned with modeling the earth's surface from aerial photographs; the second is concerned with visual interpretation for land navigation. In particular, we describe progress in the design of a core knowledge structure; representing, recognizing, and rendering complex natural and man-made objects; recognizing and modeling terrain features and man-made objects in image sequences; interactive techniques for scene modeling and scene generation; automated detection and delineation of cultural objects in aerial imagery; and automated terrain modeling from aerial imagery.

## 1 INTRODUCTION

The overall goal of the Image Understanding research program at SRI International is to obtain solutions to fundamental problems in computer vision that are necessary to allow machines to model, manipulate, and understand their environment from sensor-acquired data and stored knowledge.

In this report we describe progress in two programs.[1] The first is concerned with modeling the earth's surface from aerial photographs; the second is concerned with allowing a robotic device to successfully navigate through, and interact with, a natural 3-D environment based on real-time interpretation of sensory data.

In the discussion of the first program we describe our progress in developing techniques for automated terrain modeling from aerial imagery; automated detection and delineation of cultural objects in aerial imagery; and interactive techniques for scene modeling and scene generation.

In the discussion of the second program we describe progress in developing techniques for automated real-time recognition of terrain features and man-made objects from image sequences acquired by a combination of ranging and photographic sensors.

Common to both programs, we describe progress in developing new techniques for representing, recognizing, and rendering complex natural and man-made objects; and the construction of a core knowledge structure (CKS), which can serve as the integrating mechanism for a new generation of generic vision systems. These systems will be knowledge-base driven, rather than task-specific, using techniques in which domain knowledge is compiled into the interpretative algorithms.

[1] Supported by the Defense Mapping Agency and Defense Advanced Research Projects Agency under contracts MDA903-86-C-0084 and DACA76-85-C-0004.

## 2 DESIGN OF A CORE KNOWLEDGE STRUCTURE

The natural outdoor environment imposes significant obstacles to the design and successful integration of the interpretation, planning, navigational, and control functions of a general-purpose vision system. Many of these functions cannot yet be performed at a level of competence and reliability necessary to satisfy the needs of an autonomous robotic device. Part of the problem lies in the inability of available techniques, especially those involved in sensory interpretation, to use contextual information and stored knowledge in recognizing objects and environmental features. Our goal in this effort, described in a previous paper [Smith&Strat87], is to design a core knowledge structure that can support a new generation of knowledge-based generic vision systems.

A key scientific problem we address in this task is to devise a way of describing the appearance and characteristics of any given physical environment so thoroughly that we are assured that deficiencies in available perception (vision) techniques can be overcome by access to such prior knowledge. We cannot resort to the equivalent of using a pixel-level description as the only or ultimate solution because such detailed data would be impractical to obtain, store, retrieve, or use in the interpretive process; such low-level data would almost certainly be inaccurate when obtained, and would quickly degrade as physical changes occur. (Even illumination changes would cause a pixel-level description of appearance to become useless.) Finally, accurate interpretation must be based on more than just image appearance, and there is no immediately obvious way of describing and storing such semantic (nonpictorial) information at arbitrary levels of detail.

The CKS is designed as a community of independent interacting processes that cooperate in achieving the goals of the scene modeling system. These processes may represent sensors, interpreters, controllers, user interface drivers, or any other in-

formation processor. Each process can be both a producer and a consumer of information. Each has access to, and control over, a certain limited portion of the knowledge/database resources. The CKS architecture permits access to stored knowledge by both geographic location and by semantic content.

An initial implementation of the CKS was completed early in 1987, and subsequent activity has had two objectives: applying the CKS to a variety of distinct perception problems, and improving and extending its capabilities (based on evaluation by potential users and by lessons learned in our application efforts). In May 1987, a document specifying the initial CKS design was distributed to researchers in the Image Understanding and Strategic Computing vision communities for critical review. Most of the resulting comments focused on four aspects of the design: choice of the coordinate systems for representing spatial data, the encoding of semantic knowledge, the encoding of temporal data, and truth maintenance and the resolution of conflict. The relevant issues are briefly summarized below.

## 2.1 Coordinate Systems

A dilemma is posed in selecting a coordinate system that is used both for specifying locational information and for indexing data tokens. Is it better to use local coordinate systems in which sensed data are easily specified or should a global coordinate system be employed that forces all data into one uniform indexing system? To utilize stored spatial knowledge of the world while interpreting sensory information, there must be a means (a coordinate transformation) for relating the two. If such a transformation in fact exists, one can store and retrieve information relative to a single world-coordinate system. However, at times one may have only an approximate understanding of that transformation; consequently, to insert data into the world coordinate system when its precise position is known only locally would appear to sacrifice that precision by virtue of the uncertainty in the transformation. In the CKS, this loss of precision will be in the location of the data in the spatial directory, not in the local information stored within the data token. The spatial directory is there to facilitate retrieval, not to encode the spatial location. Consequently, the technique employed allows the data token to include whatever local coordinate information is available — yet, at the same time, the spatial retrieval mechanism is made to be uniform (albeit with a certain lack of precision) across all information sources. It should be noted that spatial reasoning should properly be applied to the data token's attributes, as these may contain the precise local information rather than the location of the token within the directory. The initial CKS implementation was concerned solely with the design and maintenance of a single world directory. Now that the initial implementation is complete, we are designing other means for specifying relative position for integration into the CKS.

## 2.2 The Semantic Directory

The semantic directory provides a means of access to the data that is orthogonal to that provided by the more conventional spatial directory. While the spatial directory allows rapid retrieval of relevant data based upon spatial location, the semantic directory provides access to the same data based upon the semantic attributes of those data. Like the spatial directory, it supports description at multiple levels of resolution -- concepts are described and data are retrieved at the level of abstraction that is most appropriate for the purpose at hand. The semantic directory also serves as a blackboard whereby one computational process can communicate with another in terms of semantic descriptions about which it is knowledgeable.

The implementation of the semantic directory consists of two components: a vocabulary of terms and a set of connections among them that serve to define their inherent relationships and properties. The vocabulary consists of a carefully chosen, domain-specific set of terms that have been identified as being both useful for, and instantiable by, the computational processes. Relationships and properties are specified by several methods. A semantic network is used to encode the specialization lattice of the concepts and the physical decomposition of composite objects. Procedural definitions are provided when it is necessary to do so. A relational database is used to retain semantic relationships that are explicitly provided. These relationships allow class properties to be inherited and alternative semantic classes to be enumerated. Each process that interacts with this knowledge base need only know how to translate to and from this vocabulary in order to share information with any other process. This avoids the need for direct translation of data between the knowledge representation of every pair of independent processes. The key motivation for this approach is to facilitate the integration of independent processes and subprocesses without restricting the representations they may use to encode their knowledge.

## 2.3 The Dynamic World

Since the world has a temporal nature, the task of building an adequate model of that world must treat time as a full variable, just as space is in the CKS. In our initial implementation of the CKS, we have not been so much concerned with modeling the dynamics present in the world as we have been with representational adequacy of the static world. However, we are pursuing a parallel effort whose specific objective is to enhance the CKS with facilities for storing, accessing, and reasoning about dynamic entities. This research has focused on two mechanisms to provide such capabilities.

1. The vocabulary and semantic network used by the CKS are being expanded to include such terms as *move*, *turn*, *grow*, *plan*, *refuel*, *traverse-route*, and *collide*. Each primitive-action schema contains slots for associated data such as time, agent, and extent. More complex procedures can be described as composites of primitive actions. This vocabulary can be used both for describing events that are taking place in the world and for representing knowledge of prototypical procedures (e.g., how to navigate around an obstacle).

2. The second mechanism involves representing dynamic objects by a volume of space swept out over a time interval. Thus a moving object can be stored in the spatial directory as the location of its track. Access routines will retrieve the object only if it satisfies the query at the specified time. A diverse collection of representations is being evaluated for its ability to provide this information efficiently for many

types of motion. Some of these are representations of physical motions, such as a cow whose movement is bounded by a fence, but is otherwise unconstrained. Still others depict movement through a conceptual space, such as the appearance of a leaf in the progression of seasons: growing, changing color, then separating from a tree.

### 2.4 Conflict Resolution and Truth Maintenance

In any database, the question of conflict resolution and truth maintenance is an important one. The CKS stores opinions that do not have to be consistent (see [Smith&Strat]). As a consequence, the CKS does not need a globally consistent database, nor is it necessary to perform truth maintenance when opinions are changed. If a process is particularly interested in knowing whether belief in its premises has changed, so that it can retract a conclusion or perform an action, it can make use of the CKS's daemon mechanism. The daemon is installed on the objects in the conclusion's support set and take the appropriate action when they are changed. This provides a process with a flexible capability for performing truth maintenance only when necessary. However, while the database does not have to be globally consistent, a process may need to recognize a conflict so that it can take appropriate action. Conflicts may occur because a single entity in the world has been described by different processes using different data tokens, or different processes may have described the same object in contradictory ways. While such descriptions are obviously related, each must be considered separately.

1. The Reference Problem — Whenever two agents (processes) desire to communicate about entities in the world, determining whether they are referring to the same entity will always be difficult. In terms of the CKS, the reference problem is having to decide whether a token used by one process refers to the same object as does a token created by another. This is a serious philosophical problem for which we have not yet found a general solution. Failure to resolve instances of the problem result in a multiplicity of tokens, an inability to draw certain inferences, and a potential for reaching false conclusions. However, we feel that, in practice, the CKS will not proliferate many tokens referring to the same objects. Rather, most sensory processes, especially vision modules, will probably be model-driven, attempting to instantiate in their imagery objects already represented in the CKS. Only when these processes discover something unexplained by prior expectations will they abandon their verification mode and create a new data token. Moreover, because a process is unable to find a token that explains that piece of data, it is unlikely that an existing token actually refers to the same object.

2. The Characterization Problem — When two processes are referring to the same object, but differ as to its semantic characterization or spatial location, we have an instance of the characterization problem. The mechanisms the CKS uses to cope with this common reference are described in the CKS specifications document [Strat&Smith87]. Using multiple opinions on the slots of data tokens is central to the CKS's design. Many difficulties are avoided because the CKS defers conflict resolution until information is retrieved. This approach has three advantages: it eliminates the resolution of conflicts that are irrelevant to the task at hand; it permits each process to take an action that the process deems appropriate; it is capable of using more appropriate information by delaying conflict resolution as long as possible. While this strategy may return more tokens as solutions to a query than would be the case if conflict resolution were performed at insertion time, it allows the CKS and its associated inference mechanisms to perform effectively even when confronted with conflicting data.

Whether one resolves conflicts at insertion or at the time of retrieval, there remains the question of whose responsibility it is to perform that function. Resolving conflicts requires knowledge of the distinctions among semantic categories as well as a detailed understanding of the capabilities and limitations of each data source. The CKS cannot bear the entire responsibility for conflict resolution because it is generally ignorant about the processes that provide the data. Furthermore, it would not be reasonable to require each sensory process to have domain knowledge enabling it to resolve all conflicts between its data and the CKS. The approach taken by the CKS is to allow processes that are capable of resolving some types of conflict to examine the data in the database, and then enter their opinions regarding such resolution into the database. Once this has been done, it is up to the user processes to decide whether to use the *resolved* opinion or carry out the analysis themselves.

Two CKS application efforts have been initiated: first, an internal effort to use the CKS in support of object recognition, planning, and navigation for an autonomous robotic device or weapon system (this work will be described in a later publication), and second, a cooperative effort with GE to use the CKS in an intelligence application (described in a separate paper by J. Mundy et al., in these proceedings).

## 3 REPRESENTING, RECOGNIZING, AND RENDERING COMPLEX NATURAL AND MAN-MADE OBJECTS

The main theoretical issue we address in this effort is how to model a large class of natural and man-made objects in a functionally useful way. The domain for our research is outdoor navigation in which a robotic device starts with an initial model of its environment and incrementally updates this model (and its relative position) as it moves to gather data or perform some task. We require the device to improve its performance by increasing its ability to recognize objects and/or decreasing its processing time as it sees things over and over again.

We have partitioned this type of perception task into three stages: model instantiation, mission planning, and execution. In the instantiation stage, a user gathers as much a priori information as possible about the area of interest. This may include selections from a standard set of cartographic items, such as terrain maps, soil classification maps, and road networks. In addition, given a specific mission, the user may interactively augment this database with higher-resolution descriptions of a few key features. In the mission-planning stage, the user explores possible vehicle paths and evaluates their viability in terms of

several factors, one of which is the interaction of the control system with the perception system. The planning stage provides such things as a list of expected landmarks and descriptions of their visibility and shape. In the third stage, the execution stage, the vehicle performs its mission, navigating around obstacles and updating its position estimates as it heads towards its objective.

A key to successful performance in all three stages is the set of representations used to describe the environment. An object may be sketched in the model instantiation stage, projected into synthetic images during the planning stage, and matched in the execution stage. Therefore, the representations, in addition to covering a wide variety of man-made and natural objects, must be able to express a range of abstraction and precision. Our strategy for exploring these representation issues is three-fold. First, we are developing a set of representations for classes of features, such as terrain patches, rocks, and trees. Second, we are developing a Core Knowledge System (see section 2) to serve as an integrating mechanism for all the information about an environment. And third, we are performing experiments using real data obtained from the "red-rock" area at the Martin-Marietta site outside Denver.

In September 1987, three SRI researchers spent a few days at Martin Marietta surveying prominent features in the selected area and gathering range and intensity data consistent with our navigation scenario. Martin-Marietta scientists modified their data acquisition programs as required and interactively drove the vehicle several times through the region to gather data. We are using these data as part of a demonstration in which we bring together several techniques produced by our longer-term investigations. In the subsections below we describe both the long-term research and its application to the demonstration task of navigating through the red-rock area.

### 3.1 Model Instantiation

The goal of the model instantiation stage is to compile as complete a model of the environment as possible prior to the definition or start of a mission. Given a specific mission, the user will interactively add mission-specific information to the environmental model. In our modeling of the red-rock region, we started with ETL's 30-meter and 5-meter digital terrain maps of the area, computed a 0.3-meter terrain model of the smaller red-rock region, and then added models of prominent discrete terrain features, such as trees and rocks. The low-resolution terrain maps provided the global context. The high-resolution map supplied a detailed ground model and key parameters for the specific object models.

We constructed the high-resolution elevation map of the red-rock region by applying a stereo technique developed by our group at SRI [Barnard88]. The resulting map provides a height estimate for each pixel in the original aerial images. For our images, a pixel corresponds to approximately one square foot on the ground. Although this map contains some errors, the majority of the heights are reasonable and at a resolution much higher than is available from any other source. The prominent rocks and trees are plainly visible.

We used the detailed height map to construct our initial model of the red-rock region, which consists of a terrain map and a set of labeled objects sitting on the terrain. We estimated the height of the terrain under large objects by interactively clipping them out of the height map and then filling the resulting holes by interpolation. We used an interactive modeling system, developed at SRI [Hanson&Quam], to build three-dimensional models of the key features, which we then entered into the CKS for permanent storage.

The individual objects are represented by superquadrics with fractal textures or as faceted volumes. They are entered in the CKS according to their semantic category and location in the world. Our initial model of the red-rock region includes about ten large trees, bushes, and rocks. In the future we plan to extend our list of semantic descriptions and develop recognition techniques that are specific to these new classes.

### 3.2 Mission Planning

The planning system has two purposes: one is to suggest and evaluate vehicle paths for accomplishing a mission; the second is to compute and "down load" mission-specific data and instructions to the vehicle control system. A typical instruction might be to aim a sensor in a certain direction and start looking for a particular object at a specific time. So far we have concentrated on the interactive evaluation of paths and have just begun to produce data and instructions for the execution-time perception system.

Our interactive evaluation system is built on top of the CKS and the TerrainCalc system, which was designed and implemented at SRI [Quam85]. It provides the user with the ability to generate sequences of images (i.e., movies) that correspond to the data that would be gathered by the vehicle's sensors if the vehicle followed the proposed path through a modeled world. These synthetic images provide a dramatic way to visualize a proposed path. The system can highlight key landmarks so the user can easily determine the range of vehicle positions from which they are visible, their range of shapes, and so on. With this system we "drive through" our model of the red-rock region and select navigational landmarks for use during the execution stage.

We have implemented projection models for two types of sensors: a conventional color camera and the ERIM range sensor. We have also implemented a model of the ERIM sensor's behavior when two surfaces at different ranges are within its beam. This level of detail is necessary to predict sensor output at occlusion boundaries and on thin objects. We have not incorporated this detailed model into the synthetic image generation because it is computationally expensive. However, in the future we plan to invoke it for special types of objects, such as thin posts or tree trunks.

### 3.3 Execution

During the execution stage, the vehicle navigates towards its destination by interpreting sensed data in terms of its predicted model of the world. To accomplish this, it performs, among other things, the following five functions: it detects unknown objects, classifies them, recognizes landmarks, tracks objects from one image to the next, and updates its world and vehicle models. Several groups, including Hughes [Daily] and Carnegie-Mellon University [Thorpe] have demonstrated techniques for detecting unknown objects in range data. However, it is significantly

harder to classify these objects into their semantic catagories (e.g., rock, bush). Classification is important for navigation because the vehicle can operate more effectively if it is done reliably. For example, the vehicle may be able to cautiously run over bushes (but not rocks). We are currently investigating ways to perform this type of object classification by using the CKS to access the semantic properties of an object and the relationships between objects.

A capability mentioned above is the updating of the world and vehicle models after the perception system has recognized and located objects. However, once a landmark has been located, it is relatively easy to track it from one image to the next. In the past, we have used three-dimensional normal distributions to represent the uncertainties associated with the world position of both the vehicle and perceived objects [Barnard86]. We are now developing a Kalman technique to maintain local vehicle-centered models in addition to the global ones.

# 4 RECOGNITION AND MODELING OF TERRAIN FEATURES AND MAN-MADE OBJECTS FROM IMAGE SEQUENCES

Our goal in this research effort is to develop automated methods for producing a labeled three-dimensional scene model from many images recorded from different viewpoints and from image sequences. We view the image-sequence approach as an important way to avoid many of the problems that hamper conventional stereo techniques because it provides the machine with previously unavailable information about the scene. The "redundant" information can be used to increase the precision of the data and filter out artifacts; the new information provided by the additional images can help to disambiguate matches for features that occur along occlusion boundaries and in the midst of periodic structures.

We have developed two techniques for building three-dimensional descriptions from multiple images. One is a range-based technique that builds scene models from a sequence of range images; the second is a motion analysis technique that analyzes long sequences of intensity images. The range technique uses data from an inertial guidance sensor on the vehicle to compenstate for vehicle attitude and position changes caused by bumps, curves, and speed changes. As a result the range data are transformed into a static world-coordinate system, which is a necessary first step for almost all further analysis. By combining the data from multiple images, we are able to filter out artifacts and produce a more complete map of the region in front of the vehicle. We have developed several representations of these three-dimensional data, including height maps, orientation images, and voxel arrays, each of which offers distinct coherence and resolution advantages to the analysis procedures.

In the past we have developed techniques that analyze these representations of the range data to identify such features as support surfaces, ditches, and thin raised objects. We are currently extending these techniques to track such objects through sequences of images and gradually transform their shape and position models. As part of this updating procedure, we are exploring the trade-offs associated with alternative matching procedures. In one method we apply extensive bottom-up analysis to build as complete a description as possible of each image before attempting recognition. In another we predict as much as possible from one image to the next and attempt to limit the low-level analysis to key portions of the images. Our goal is to determine the appropriateness of these techniques, and variations of them, as a function of such things as the accuracy of the inertial navigation information, the type of object being recognized, and the reliability of the bottom-up analysis.

In [Bolles,Baker,&Marimont87] we presented a motion analysis technique, which we call Epipolar-Plane Image (EPI) Analysis. It is based on considering a dense sequence of images as forming a solid block of data. Slices through this solid at appropriately chosen angles intermix time and spatial data in such a way as to simplify the partitioning problem: These slices have more explicit structure than the conventional images from which they were obtained. In the paper we demonstrated the feasibility of this novel technique for building structured, three-dimensional descriptions of the world. Recently we have extended this technique (see [Baker&Bolles88]) to locate surfaces in the spatio-temporal solid of data, instead of analyzing slices, in order to maintain the spatial continuity of edges from one slice to the next. In addition, we have reimplemented the analysis to work incrementally, applying a Kalman filter to update the three-dimensional description of the world each time a new image is received. As a result of these changes the program produces extended three-dimensional contours instead of sets of isolated points. These contours evolve over time. When a contour is initially detected, its location is only coarsely estimated. However, as it is tracked through several images, its shape typically changes into a smooth three-dimensional curve that accurately describes the corresponding feature in the world. In the future we plan to develop descriptions of the surfaces between the contours, explore a parallel implementation of our surface-building algorithm, and apply it to other types of data, such as computed tomography or magnetic resonance imaging data.

# 5 INTERACTIVE TECHNIQUES FOR SCENE MODELING AND SCENE GENERATION

Manual photointerpretation is a difficult and time-consuming step in the compilation of cartographic information. On the other hand, fully automated techniques for this purpose are currently incapable of matching the human's ability to employ background knowledge, common sense, and reasoning in the image-interpretation task. Near-term solutions to computer-based cartography must include both interactive extraction techniques and new ways of using computer technology to provide the end-user with useful information in a primarily iconic, rather than symbolic, format.

In order to support research in semiautomated and automated computer-based cartography, we have developed the SRI Cartographic Modeling Environment. In the context of this interactive workstation-type system, the user can manipulate multiple images; camera models; digital terrain elevation data; point, line, and area cartographic features; and a wide assortment of three-dimensional objects. Interactive capabilities include free-hand feature entry and editing, altering features while

constraining them to conform to the terrain and lighting geometry, and adjustment of the scene viewpoint. Synthetic views of a scene from arbitrary viewpoints may be constructed using terrain and feature models in combination with texture maps acquired from aerial imagery. This ability to provide an end-user with an interactively controlled scene-viewing capability could eliminate the need to produce hard-copy (symbolic) maps in many application contexts. Additional applications include high-resolution cartographic compilation, direct utilization of cartographic products in digital form, and generation of mission-planning and training scenarios.

A summary of our progress in this area is described in two papers: the basic design issues for this system can be found in [Hanson,Pentland,&Quam87]. An overview of the current implementation is presented in a separate paper in these proceedings, [Hanson&Quam88].

# 6 AUTOMATED DETECTION AND DELINEATION OF CULTURAL OBJECTS IN AERIAL IMAGERY

The detection, delineation, and recognition of any significantly broad class of objects (e.g., buildings, airports, cultivated land) in aerial imagery has proven to be an extremely difficult problem. In fact, a nominal component in the solution of this problem, image partitioning, is considered to be one of the most refractory problems in machine vision.

We have recently formulated an optimization-based approach, applicable both to image partitioning and to subsequent steps in the scene analysis process, that involves finding the "best" description of the image in terms of some specified descriptive language.

In the case of image partitioning [Leclerc88], we employ a language that describes the image in terms of regions having a low-order polynomial intensity variation plus white noise; region boundaries are described by a differential chain code. A "continuation" technique is used to find a "best" description, in the sense of least encoding length, that is both stable (i.e., minor perturbations in the viewing conditions should not alter the description) and complete (i.e., the image, including any noise or errors, must be completely explained by the description).

In situations where the required image description must proceed beyond that of a delineation of coherent regions, we require an extended vocabulary relevant to the semantics of the given task. [Fua&Leclerc88] deal with the problem of boundary/shape detection given a rough estimate of where the boundary is located and a set of photometric (intensity-gradient) and geometric (shape-constraint) models for a given class of objects. They define an energy (objective) function that assumes a minimal value when the models are exactly satisfied. An initial estimate of the shape and location of the curve is used as the starting point for finding a local minimum of the energy function by embedding this curve in a viscous medium and solving the dynamic equations. This energy-minimization technique, which evolved from a less-efficient gradient-descent approach, has been applied to straight-line boundary models and to more complex models that include constraints on smoothness, parallelism, and rectilinearity. It has also been incorporated into the SRI Cartographic Modeling Environment described earlier. In an interactive mode, the user supplies an initial estimate of the boundary of some object (which may be quite complex, like the outline of a tree) and then, if need be, corrects the optimized curve by applying forces to the curve or by changing one of a few optimization/model parameters.

For the important cartographic task of extracting cultural features and vegetation from aerial imagery, [Fua&Hanson88] describe a technique which involves optimizing the evidence for model instances in the image data. As basic model components are assembled into more complex structures, semantic constraints are used to optimize the state of the model and to hypothesize missing model components in the image itself. This approach to shape extraction contrasts strongly with current model-based vision techniques that rely on passive interpretation of precomputed syntactic image features. Possible parses of a scene are ranked using a measure related to the cost of encoding the scene information in terms of (languages of) generic models. As examples of generic shapes, models of buildings, roads and vegetation clumps are introduced. The model definitions include edge characteristics, two- and three-dimensional edge and face relationships, region characteristics, and procedures for predicting and discovering missing shape components.

We believe that both the Leclerc and the Hanson and Fua techniques represent significant advances in the state-of-the-art in their respective areas of image partitioning and delineation of cultural features. Both systems have been able to produce excellent results in complex situations where existing (typically local) approaches fail. Future work (both systems) involves vocabulary extensions and devising more effective optimization procedures.

# 7 AUTOMATED TERRAIN MODELING FROM AERIAL IMAGERY

Stereo reconstruction is a critical task in cartography that has received a great deal of attention in the image understanding community. Its importance goes beyond its obvious application to constructing geometric models: Understanding scene geometry is necessary for effective feature extraction and other scene analysis tasks. While considerable success has been achieved in important parts of the problem, there is no complete stereo-mapping system that can perform reliably in a wide variety of scene domains.

The standard approach to the problem of stereo mapping involves finding pairs of corresponding scene points in two images (which depict the scene from different spatial locations) and using triangulation to determine scene depth. Various factors associated with viewing conditions and scene content can cause the matching process to fail; these factors include occlusion, projective or imagining distortion, featureless areas, and repeated or periodic scene structures. Some of these problems can only be solved by providing the machine with more information, which may take the form of additional images or descriptions of the global context.

Our research strategy in this task is to develop new techniques for the key steps in the stereo process, such as matching and interpolation, and, in parallel, to investigate ways to integrate these new ideas with existing techniques. As part of this process we have implemented [Hannah85] and evaluated [Hannah88] a

complete high-performance stereo system. In a test of existing stereo systems on 12 pairs of digital images, conducted by the International Society of Photogrammetry, Hannah's system was able to successfully process more of the images than any other system (11 out of the 12 pairs); while full evaluation of the test results is still not complete, it appears that her system will rank first (or very near the top) in the competition.

We are currently investigating two novel approaches to stereo depth recovery, which are significant departures from the conventional paradigm and which have important implications for other problems in scene analysis.

G.B. Smith developed a method for dense stereo compilation that entirely avoids local matching [Smith85 and Smith86]. The procedure begins with stereo images assumed to be in correspondence so that depth recovery can be accomplished for individual scan lines (i.e., the horizontal scan lines in the two images are corresponding epipolar lines). Smith showed how to set up systems of simultaneous equations whose solution is the depth profile corresponding to the intersection of the epipolar plane with the scene surfaces. Experiments with synthetically generated scenes show that the technique is theoretically sound, but the approach requires further work because in its present implementation it is overly sensitive to noise.

Barnard developed a stereo compilation technique that embeds local matching, at the level of individual pixels, in a global optimization framework [Barnard86 and Barnard87]. His objective function rewards correspondences between pixels that are similar in intensity value and that imply disparities that are similar to those of their neighboring pixels. Simulated annealing is used to find a complete set of correspondences that best satisfies the objective function. Because individual pixels (rather than finite areas) are matched, projective distortion is no longer a problem, nor does one have to worry about adjusting the size or shape of a correlation patch. Experiments show that this approach can successfully compile a dense depth model of natural three-dimensional (ground-level) scenes.

In his paper in these proceedings, Barnard describes an improved version of his stochastic stereo matching system. He sets up an analogy between his objective function and the potential energy of a system consisting of a lattice of spring-loaded, spring-coupled oscillators. Simulated annealing is used find approximations to the ground states (i.e., the states of minimal energy) of the physical system. The approach adapts a new technique from statistical physics (microcanonical simulation) that has several advantages over standard (canonical) simulated annealing. It can be easily implemented with only integer arithmetic, does not require the evaluation of $exp(x)$, does not require high-quality random numbers, and is considerably more efficent. The matcher also uses a coarse-to-fine approach for the efficient matching of large images with substantial ranges of disparity. Ground states are first approximated at coarse scales. The map is then transformed to a finer scale and another cycle of annealing is performed. This involves heating the system to a high temperature, which may be done relatively quickly, and then slowly cooling it to the finer-grained ground state. (The heating phase is needed to allow information to diffuse through the lattice.) Annealing provides a way to bridge the gap between scales. The emphasis in this work has been to develop a practical, efficient method for dense stereo matching over a wide variety of scenes. (As noted earlier, we have already used this technique to produce dense elevation maps of the ALV test site – we can map on 0.3 meter centers as compared with 5 meters in the best previously available DTMs.)

While the stereo problem will remain a focus of a portion of our research effort, our main concern is to develop an understanding of how knowledge of scene depth information can be effectively used in the scene-partitioning and object-recognition tasks.

# 8 ACKNOWLEDGMENT

The following researchers have contributed to the work described in this report: H.H. Baker, S.T. Barnard, A. Bobick, R.C. Bolles, O. Firschein, M.A. Fischler, P. Fua, M.J. Hannah, A.J. Hanson, K.I. Laws, Y. Leclerc, D. Marimont, A.P. Pentland, L.H. Quam, G.B. Smith, T.M. Strat, L. Wesley, and H.C. Wolf.

# 9 REFERENCES

Recent publications resulting fully or in part from our image-understanding research program:

1. Baker, H.H., "Building Surfaces of Evolution: The Weaving Wall," DARPA Image Understanding Workshop, Washington, D.C., *(this proceedings)*, April 1988.

2. Baker, H.H. and R.C. Bolles, "Generalizing Epipolar-Plane Image Analysis on the Spatiotemporal Surface," DARPA Image Understanding Workshop, Washington, D.C., *(this proceedings)*, April 1988.

3. Baker, H.H., R.C. Bolles, and D.H. Marimont, "Generalizing Epipolar-Plane Image Analysis for Non-Orthogonal and Varying View Directions," Proc. Image Understanding Workshop, University of Southern California, Vol.II, pp.843-848, February 1987.

4. Barnard, S.T., "Stochastic Stereo Matching Over Scale," DARPA Image Understanding Workshop, Washington, D.C., *(this proceedings)*, April 1988.

5. Barnard, S.T., "Stereo Matching by Hierarchical, Microcanonical Annealing," Proc. Image Understanding Workshop, University of Southern California, Vol.II, pp.792-796, February 1987.

6. Barnard, S.T., R.C. Bolles, D. Marimont, and A.P. Pentland, "Multiple Representations for Mobile Robot Vision," Proc. SPIE Symposium on Advances in Intelligent Robotics Systems, Cambridge, Massachusetts, October 26-31, 1986.

7. Barnard, S.T., "A Stochastic Approach to Stereo Vision," Proc. 5th National Conf. on Artificial Intelligence, Philadelphia, Pennsylvania, pp.676-680, August 11-15, 1986.

8. Bolles, R.C., H.H. Baker, and D.H. Marimont, "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion," *International Journal of Computer Vision*, Kluwer Academic Publishers, Vol.I, No.1, pp.7-5, June 1987.

9. Bolles, R.C. and H.H. Baker, "Epipolar Image Analysis: A Technique for Analyzing Motion Sequences," Proc. 3rd. Int. Symp. on Robotics Research, Paris, France, October 1985.

10. Daily, M.J., J.G. Harris, and K. Reiser, "Detecting Obstacles in Range Imagery," in the Proceedings of the Image Understanding Workshop, pp. 87-92, February 1987.

11. Fischler, M.A. and R.C. Bolles, "Image Understanding Research at SRI International," Proc. Image Understanding Workshop, University of Southern California, Vol.I, pp.12–17, February 1987.

12. Fischler, M.A. and R.C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," Comm. ACM, Vol.24(6), pp.381–395, June 1981.

13. Fischler, M.A. and O. Firschein, "Intelligence: the Eye, the Brain, and the Computer," Addison-Wesley, Reading, Massachusetts, 1987.

14. Fischler, M.A. and O. Firschein, "Readings in Computer Vision," Morgan-Kaufmann, Los Altos, California, 1987.

15. Fischler, M.A. and O. Firschein, "Parallel Guessing: A Strategy for High-Speed Computation," Pattern Recognition, Vol.20, No.2, pp.257–263, 1987.

16. Fua, P.V. and A.J. Hanson, "Extracting Generic Shapes Using Model-Driven Optimization," DARPA Image Understanding Workshop, Washington, D.C., *(this proceedings)*, April 1988.

17. Fua, P. and A.J. Hanson, "Using Generic Geometric Models for Intelligent Shape Extraction," Proc. Image Understanding Workshop, University of Southern California, Vol.I, pp.227–233, February 1987.

18. Fua, P. and A.J. Hanson, "Using Generic Geometric Knowledge to Delineate Cultural Objects in Aerial Imagery," Tech. Note 378, Artificial Intelligence Center, SRI International, Menlo Park, California, March 1986.

19. Fua, P. and A.J. Hanson, "Resegmentation Using Generic Shape: Locating General Cultural Objects," Pattern Recognition Letters, in press, 1986.

20. Fua, P. and A.J. Hanson, "Locating Cultural Regions in Aerial Imagery using Geometric Cues," Proc. DARPA Image Understanding Workshop, Miami Beach, Florida, pp.271–278, Dec. 9-10, 1985.

21. Fua, P.V. and Y.G. Leclerc, "Model Driven Edge Detection," DARPA Image Understanding Workshop, Washington, D.C., *(this proceedings)*, April 1988.

22. Fua, P. and Y. Leclerc, "Finding Object Boundaries Using Guided Gradient Ascent," Proc. Image Understanding Workshop, University of Southern California, Vol.II, pp.888–891, February 1987.

23. Hannah, M.J., "Test Results from SRI's Stereo System," DARPA Image Understanding Workshop, Washington, D.C., *(this proceedings)*, April 1988.

24. Hannah, M.J., "Evaluation of STEREOSYS vs Other Stereo Systems," Technical Note 365, Artificial Intelligence Center, SRI International, Menlo Park, California, October 1985.

25. Hannah, M.J., "The Stereo Challenge Data Base," Technical Note 366, Artificial Intelligence Center, SRI International, Menlo Park, California, October 1985.

26. Hannah, M.J., "SRI's Baseline Stereo System." Proc. DARPA Image Understanding Workshop, Miami Beach, Florida, pp.149–155, Dec. 9-10, 1985.

27. Hanson, A.J. and L. Quam, "Overview of the SRI Cartographic Modeling Environment," DARPA Image Understanding Workshop, Washington, D.C., *(this proceedings)*, April 1988.

28. Hanson, A.J., A.P. Pentland, and L.H. Quam, "Design of a Prototype Interactive Cartographic Display and Analysis Environment," Proc. Image Understanding Workshop, University of Southern California, Vol.II, pp.475–482, February 1987.

29. Heeger, D. J., "Optical Flow from Spatiotemporal Filters," Proc. First International Conference on Computer Vision, London, England, pp. 181-190, June 8-11, 1987.

30. Laws, K.I., "Goal-Directed Textured-Image Segmentation," Proc. SPIE Conf. on Applications of Artificial Intelligence II, Vol. 548, Arlington, Virginia, April 9-11, 1985.

31. Laws, K.I, "Experiments in Navigational Road Tracking," in S.J. Rosenschein, M.A. Fischler, and L.P. Kaebling, "Research on Intelligent Mobile Robots," Final Technical Report, Project 7390, SRI International, Menlo Park, California, pp. 151–157, May 20, 1986.

32. Leclerc, Y., "Constructing Simple Stable Descriptions for Image Partitioning," DARPA Image Understanding Workshop, Washington, D.C., *(this proceedings)*, April 1988.

33. Leclerc, Y., "Capturing the Local Structure of Image Discontinuities in Two Dimensions, " Proc. IEEE Conf. on Computer Vision and Pattern Recognition, San Francisco, California, pp. 34–38, June 19-23, 1985.

34. Marimont, D.H., "Projective Duality and the Analysis of Image Sequences," Proc. of the Workshop on Motion: Representation and Analysis, IEEE Computer Society, pp. 7–14, Kiawah Island, South Carolina, May 1986.

35. Pentland, A.P., "Fractal-Based Description of Natural Scenes," IEEE PAMI 6(6):661-674, 1984.

36. Pentland, A.P., "A New Sense for Depth of Field," Proc. 9th Int. Joint conf. on Artificial Intelligence, Los Angeles, California, pp. 988–994, August 18-23, 1985.

37. Pentland, A.P. (ed.), "From Pixels to Predicates," Ablex, Norwood, New Jersey, 1985.

38. Pentland, A.P., "On Describing Complex Surfaces," Image and Vision Computing, 3(4):153-162, November 1985.

39. Pentland, A.P., "Part Models," Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Miami Beach, Florida, pp. 243–249, June 22-26, 1986.

40. Pentland, A.P., "Parts: Structured Descriptions of Shape," Proc. 5th National Conf. on Artificial Intelligence, Philadelphia, Pennsylvania, pp. 695–701, August 11-15, 1986.

41. Pentland, A.P., "Perceptual Organization and Representation of Natural Form," Artificial Intelligence, 28:293-331, 1986.

42. Pentland, A.P., "Recognition by Parts," SRI Technical Note 406, Dec. 1986.

43. Quam, L.H., "The TerrainCalc System," Proc. DARPA Image Understanding Workshop, Miami Beach, Florida, pp. 327–330, Dec. 9-10, 1985.

44. Smith, G.B. and T.M. Strat, "Information Management in a Sensor-Based Autonomous System," Proc. Image Understanding Workshop, University of Southern California, Vol.I, pp. 170–177, February 1987.

45. Smith, G.B., "Stereo Reconstruction of Scene Depth," Proc. Computer Vision and Pattern Recognition, San Francisco, California, pp. 271–276, June 19-23, 1985.

46. Smith, G.B., "Stereo Integral Equation," Proc. 5th National Conf. on Artificial Intelligence, Phildelphia, Pennsylvania, pp. 689–694, August 11-15, 1986.

47. Strat, T.M. and G.B. Smith, "Core Knowledge System: Storage and Retrieval of Inconsistent Information," DARPA Image Understanding Workshop, Washington, D.C., *(this proceedings)*, April 1988.

48. Strat, T.M. and M.A. Fischler, "One-Eyed Stereo: A General Approach to Modeling 3-D Scene Geometry," Proc. 9th Int. Joint Conf. on Artificial Intelligence, Los Angeles, California, pp. 937–943, August 18-23, 1985.

49. Thorpe, C., S. Shafer, and T. Kanade, "Vision and Navigation for the Carnegie Mellon Navlab," in the Proceedings of the Image Understanding Workshop, pp. 143-152, February 1987.

50. Wesley, L.P., "Evidential Knowledge-Based Computer Vision," Optical Engineering, 25(3):363-379, March, 1986.

# Summary of Image Understanding Research at the University of Massachusetts[1]

Edward M. Riseman and Allen R. Hanson

**Computer and Information Science Department**
**University of Massachusetts at Amherst**

**ABSTRACT**

This paper summarizes several areas of research at the University of Massachusetts that are partially or entirely supported under the DARPA Image Understanding Program. Many of the individual efforts discussed below are further developed in other papers in these proceedings. The summary is divided into several areas:

1. Knowledge-Based Vision
2. Database Support for Symbolic Vision Processing
3. Motion Processing
4. Perceptual Organization (Grouping)
5. Image Understanding Architecture
6. Integrated Vision Benchmark for Parallel Architectures
7. Mobile Vehicle Navigation.

Although we discuss each area separately, a fundamental goal of the computer vision research environment at UMass is the integration of a diverse set of research efforts into a system that is ultimately intended to achieve real-time image interpretation. Two of our major system integration efforts are the VISIONS static interpretation system, which is a knowledge-based computer vision system utilizing parallel modular processes that communicate via a blackboard [DRA87b,HAN87a]. The second system integration effort involves an autonomous mobile vehicle for navigating through a partially known environment [ARK87a,ARK87b,ARK87c].

## 1 Knowledge-Based Vision

### 1.1 VISIONS Scene Interpretation

Research in knowledge-based vision has continued via the development of the VISIONS Schema System and its application to a variety of domains, including road scenes, house scenes, and aerial images [DRA87a,DRA87b,HAN87a,REY87]. Integrating concepts from artificial intelligence and computer vision, the Schema System provides a frame work for building prototype application systems employing a knowledge-based approach

[illegible]

A primary characteristic of this work is the utilization of coarse-grained parallelism at the semantic level, where modular processes called schemas provide object-specific control strategies for recognition of object instances in expected contexts. Schemas partition both knowledge and computation in terms of natural object classes for a given domain; (note that the term "object" is used loosely to include object parts, objects, and scene contexts comprised of sets of objects). Each object class has a corresponding schema which contains declarative and control knowledge specific to that class. To identify an instance of the object class in an image, a schema instance is created, which is an executable copy of the schema associated with the object class. One schema instance is invoked for each object instance hypothesized to be in the scene.

Schemas run as independent concurrent processes. Communication between schema instances is carried out asynchronously by means of a global blackboard through which schemas cooperate and compete to identify and locate the significant objects present in the scene. Consistency is enforced by the requirement that each spatial area of the image is to have only one interpretation, and that all such local interpretations are to be mutually compatible.

There is now a serious effort getting underway to bring the Schema System up on a shared-memory multiprocessor (a Sequent) to begin to explore these ideas on a real machine. The issues here significantly overlap much of what appears in the remaining areas of this summary since those systems will be the components controlled by the interpretation system. In particular this research effort complements that of the Image Understanding Architecture (IUA) [WEE88a,WEE88b], since only a vertical slice of the IUA is being constructed with a single symbolic processor at the the top of the three-level architecture. Thus, the multiprocessor investigation of schemas and the experimental experience of applying it should give insight into the top-level design of the IUA before any attempt is made to scale it up (see section 5. of this summary).

### 1.2 Object Recognition Using 3D Models

Our primary concern with the schema approach has been on control issues in the parallel recognition of scenes and objects. The experimental knowledge bases constructed to date have utilized primarily two-dimensional processes, with only limited use of three-dimensional information. However, the importance of three-dimensional geometric models is quite obvious. Burns and Kitchen [BUR87a,BUR87b,BUR88] are developing an object recognition system that is designed to handle the computational complexity posed by a large model base of geometric object models, an unconstrained viewpoint, and the two dimensional

structural detail that will appear in the various views of an object.

The design is based on two ideas. The first is that before recognition, in an off-line process, the three-dimensional model base can be precompiled into descriptions, called predictions, of the potential appearance of all objects from all viewpoints. This reduces the recognition process during interpretation to a 2D matching process and avoids the more complex 3D-to-2D transformations between 3D models and 2D images. The second is to represent all the predictions in a prediction hierarchy. The nodes in this hierarchy are partial 2D descriptions that are common to multiple object views and hence constitute shared processing subgoals during matching. Thus, at any stage of extracting 2D geometric structure, the prediction hierarchy provides an indexing path to the 3D models that might have produced them.

A current focus of this work is on the automatic compilation of a prediction hierarchy from a 3D model base. A prototype system using a set of polyhedral objects and projections from an unconstrained range of viewpoints is under development [BUR88].

### 1.3 Goal-Directed Control of Low-Level Processes for Image Interpretation

In a recently completed thesis, Kohl [KOH88,KOH87] adopted the view that the task of image interpretation should be viewed as a coordinated process in which high-level interpretation strategies and low-level image segmentation processes interact through the intermediate-level of processing. He developed a system called GOLDIE (Goal-Directed Intermediate Level Executive) which mediates this interaction and which provides top-down control over low- and intermediate-level processes. Requests for low-level processing are represented in the form of a goal, which contains in its structure constraints on the form of the resulting data. Posting a goal to GOLDIE results in the creation of an intermediate level process which is an instantiation of a schema control strategy for achieving the goal. Thus, the set of schemas represented at the intermediate level define the types of goals which may be processed.

The active schemas utilize knowledge of the image domain as well as measurements on the image itself to translate the goals into appropriate low-level process specifications, which include the identification of the image features, algorithms, and algorithm parameters to be used by the system to accomplish the task. The low-level process specifications are then executed by a process controller, which returns the results to the intermediate-level control process. The constraints on the results are then checked and, if satisfied, the results are returned to the requesting high-level schema. If the results are unsatisfactory, a new process specification is generated and the cycle repeats until all possible avenues are exhausted. Execution of a process may result in the generation of subgoals, causing a recursive invocation of GOLDIE.

Although GOLDIE was developed primarily to control the process of region segmentation, the development of the segmentation incorporates both edge and line information. In particular the initialization schema of GOLDIE, which is executed at system startup in the absence of high level goals, uses both representations and has proven to be a very effective region segmentation algorithm. Future versions of GOLDIE will extend the schema-based control paradigm to include all of the low and intermediate level algorithms currently in use.

### 1.4 Information Fusion

A constraint based approach to uniformly combining information from multiple representations and sources of sensory data has been developed [BEL86,RIS87]. The approach is important to research in intermediate level grouping, knowledge-based model matching, and information fusion. The techniques extend the capabilities of an earlier system [HAN87b] that applied constraints to attributes of single types of extracted image events, called tokens. Relational measures are defined between symbolic tokens so that subsets of tokens across representations can be selected and grouped on the basis of constraint functions applied to these relational measures.

Since typical low-level representations involve hundreds or thousands of tokens in each representation, even binary relational measures can involve very large numbers of token pairs. Control strategies for ordering and filtering tokens, based upon constraints on token attributes and token relationships, can be formed to reduce the computation involved in producing token aggregations. The system was demonstrated using region and line data and a simple set of relational measures. The approach can be naturally extended to include tokens extracted from motion, stereo, and range data.

### 1.5 Evidential-Based Control in Knowledge-Based System

Wesley [WES82,WES83,WES86,WES88] takes a somewhat different view of control in knowledge-based systems. Arguing that such systems which operate in real-world environments must necessarily reason about their actions from information that is inherently uncertain, impecise, and occasionally incorrect, he develops a system called OCULUS in which control related information is viewed as evidence for and against hypothesized control decisions. Using the Shafer-Dempster theory of evidence [DEM68,SHA76] as the mathematical foundation for evidential reasoning, OCULUS derives control related evidence from control knowledge sources which make measurements on the state of a knowledge-based interpretation system. Evidential decision measures were developed to help choose an action based upon the results of the inference process.

A simulated vision environment was constructed using results of the VISIONS system interpretation. Simulated KSs were parameterized via characteristics such as accuracy and ambiguity. Wesley's results demonstate that an evidential approach to control can improve OCULUS's ability to correctly interpret images by as much as 30% in most cases, with less than a 10% increase in effort. Furthermore, the system is shown to degrade gracefully as the quality of the information supplied by the control knowledge sources decreases. Wesley suggests that the domain independent control strategies developed in OCULUS might be effective in knowledge-based systems that operate in other real-world task domains.

## 2 Database Support for Symbolic Vision Processing

It is becoming increasingly evident that intermediate-level vision, and the perceptual grouping processes encompassed, are

an extremely important component of any knowledge-based interpretation system. Our current view is that a major goal of the perceptual organization processes is to reduce the substantial gap which exists between the extracted image descriptors and the high level knowledge representations of the objects. The more abstract the intermediate level tokens are, the more computationally efficent the matching is between high level descriptions and the intermediate level tokens, where general world knowledge is used to constrain the set of possible interpretations.

The intermediate level may be viewed as simply a symbolic representation of primitive image 'events' as points, regions, lines, contours, areas, surfaces, etc. and their features, created by an iconic to symbolic transformation of the image data. However, recent work in vision has shown it to be much more than a passive level of data representation. Many of the recently developed grouping operators, for example, function at the intermediate level by building more abstract structures from the primitive descriptions [BOL87,DOL88,DOL86,FIS86,LOW85,REY87,WIL88b,WIT83].

Consequently, we view the intermediate level as hosting active processes which construct more abstract tokens from less abstract ones. Universally applicable similarity operators and geometric constraints are employed on the evolving spatial structures. In order to facilitate research on image interpretation systems, where data and control are closely coupled throughout all three stages, mechanisms must be provided for efficient structuring of the data and processes.

In addition, the complexity of many vision systems requires the cooperation and interaction of many researchers and the integration of their subsystems. The applications are far too large for an individual to solve on his own. Thus, the intermediate level representations and software environment must support, at a minimum, the following:

- a single uniform data interface to both high and low levels;
- sharing of data between levels, and between researchers at all levels;
- integration of research results into a monolithic system;
- standard handling of common relational and geometric queries, to reduce the programming overhead of coding them from scratch;
- distribution of data and processes over several machines and in several computer languages (C, LISP, FORTRAN)
- an efficient programming environment for intermediate level algorithm development.

Unfortunately, current understanding of this level of vision makes it impossible to predict the kind of structures which must be represented, the types of access to these structures, the kinds of relationships which might exist between them, or the range and type of descriptive features attached to them. At this point it appears that quite a diverse set of representations and mechanisms are employed in various vision system components. We can minimally assume that the intermediate level must support known methods of information fusion and perceptual organization, and provide the flexibility to support the representation and manipulation of geometric and structural relations. For example, the types of data which should be representable at this level include

- points: endpoints, points of high curvature, vertices, virtual points, etc;
- lines and curves: edges, straight lines, curve segments;
- areas: regions, surface patches, focus of attention areas, etc.;
- relations: adjacency, containment, intersection, etc;
- structures: grouped lines and edges, edge-vertex tuples (e.g.corners), line chains, geometric structures, and generally subsets of tokens defined by a relation.

Each has an associated set of features, or descriptors, whose definition may vary as research progresses. Consequently, there are two fundamental types of data access that must be supported: access to tokens by name and by feature value (associative access); note that we also treat relations as features. It is rarely the case that a token definition stays constant over the course of an interpretation. Tokens may be split or merged with other tokens, features recomputed, and tokens may take part in many set relationships with other tokens.

## 2.1 ISR1

Research into intermediate level grouping mechanisms [BOL87,DOL88,FIS86,LOW85,REY87,WIL88b,WIT83] and the development of the VISIONS schema system [DRA87a,DRA87b,DRA88,HAN78a,HAN78b,HAN78c,HAN86] [HAN87a] have led us towards the development of a flexible and efficient intermediate level of representation called the Intermediate Symbolic Representation (ISR) [BRO88,DRA87a,HAN87a].

ISR1 was implemented in 1985 primarily as a data interface between the output of the low-level image segmentation and feature extraction processes running in C on a DEC VAX and the high-level symbolic interpretation system running in Lisp on a TI Explorer. The unit of representation in ISR1 is the token, composed of a name and a list of features. The features are described through a lexicon and tokens sharing a common lexicon are organized into a tokenset. Each feature entry in the lexicon consists of a datatype and an optional on-demand function for computing the feature value. Standard feature datatypes include type real, integer, pointer, extents, and bitplane. Extents is simply the coordinates of the bounding rectangle of the token in the image plane. Features of type bitplane are binary masks defining the spatial coverage of the token in the image.

Since a tokenset may be viewed as a two-dimensional array, access to elements in the array are by token name (the rows) and constraints on feature values (the columns). Associative access of elements are returned as a list or as an array. One of the major design deficiencies of ISR1 was that there were no convenient mechanisms for representing and storing these elements.

Standard I/O and file handling utilities are provided for creating libraries of tokensets and lexicons. File transfer is a common method of communication among the different processors in our environment.

## 2.2 ISR2

ISR1 was used heavily over a period of years by researchers whose individual research focus was distributed reasonably uniformly over all three levels of abstraction. During this period of

time, a number of design deficiencies were noted in ISR1; two of the major problems which necessitated the redesign were:

- The separation of the lexicon from a tokenset created problems. When the lexicon had to be modified, old tokensets no longer had valid descriptions; (for example, when a feature was added to a set of region tokens). Short-term solutions resulted in a proliferation of stored tokensets and a great deal of confusion at the application level.

- Sets of associatively accessed tokens could not be conveniently manipulated, made into tokensets, nor stored as features of other tokens. In particular, it was difficult to relate tokens across token types (such as regions and lines).

In response to these problems a decision was made to design a new version of the symbolic database [BRO88]. ISR2 retains the basic flavor of ISR1, including tokensets, the basic token access functions, and features and feature datatypes. The lexicon concept was eliminated in favor of associating the feature descriptions with the tokenset itself. Recognizing that there were other pieces of information which apply to the tokenset as a whole (such as generation dates, image information, and processing history), each tokenset is now organized as a simple frame, with slots for the various features of the tokenset. Frame features include the simple types integer, real, string, frame, and tokenset and the complex types composite, sort, slice, and virtual. The frame feature allows frame hierarchies to be constructed. The tokenset feature points to the tokenset or tokensubset associated with the frame. The composite feature is a generalization of feature types like bitplane and extents from ISR1 (i.e. they are multi-valued features). Virtual features are features whose values can be calculated but not stored, hence they are always calculated on demand. They serve much the same purpose as methods in an object-oriented programming language. Sort and slice datatypes provide facilities for defining and maintaining partitions based on feature values; for example, a typical application for a slice feature might be to create and maintain a grid for fast 2D spatial access to tokens from the image coordinates. Other modifications to ISR1 include a more comprehensive file management system to deal with the frame hierarchies, the addition of several types of demons (on-demand functions), and extensions to the command language to support the new capabilities.

Like ISR1, much of ISR2 will be implemented in C with a LISP user interface. Implementation is expected to begin in the near future. Since vision is such a dynamic research environment, it would not be unreasonable to expect ISR3 after experience with ISR2 is obtained.

# 3 Motion

## 3.1 Background

In the area of motion analysis, research has continued down several avenues: a subsystem for extracting depth from approximate translational motion for the CMU NAVLAB, an algorithm for binocular motion analysis, a closed form solution for recovery of general motion parameters under assumptions of constant motion, and an algorithm for spatio-temporal grouping and tracking of linear structures and recovery of their associated depth.

Over the last 7 years, our research group has developed a variety of motion algorithms, and in most cases applied them to real-world image sequences, including domains of robot arm workspaces, indoor hallways, and outdoor sidewalk/road scenes. In particular, experimental investigations of translational motion sequences demonstrated some degree of robustness. Anandan [ANA85b,ANA87a,ANA87b,ANA88] developed an algorithm for determining feature point correspondences between frames that allowed the computation of dense displacement fields with associated confidences. This capability could be used to effectively track points across frames. Lawton [LAW83,LAW84] showed that the focus-of-expansion (FOE) often could be extracted from a sensor undergoing pure translational motion (i.e two degrees of freedom) to within a few degrees of accuracy. Glazer [GLA87a,GLA87b], in his recently completed Ph.D. thesis, developed two algorithms for the efficient computation of image motions using hierarchical multiresolution methods operating over image data pyramids.

Bharwani [BHA85,BHA86] developed a multi-frame algorithm for depth extraction under known translational motion which iteratively predicts the image motion of a feature point in future frames, determines correspondence by a search over the limited predicted area, and then refines the depth estimate using the new match. Snyder [SNY86] analyzed the effects of uncertainty in the location of the FOE and feature points in the image on the computation of depth, and showed how this analysis could be used to quantitatively provide predictions for constraining the search window used for matching these points in future frames.

Adiv [ADI85a,ADI85b,ADI85c] developed an algorithm for general sensor motion (five degrees of freedom) in an environment with objects undergoing independent general motion, the goal being to recover the motion parameters of both the sensor and any visible moving objects. This latter problem is much harder, and although there was some empirical demonstration of capabilities, there was an assumption that this algorithm would be computationally more complex, and perhaps less robust, than the algorithms for translational motion.

In this volume research is presented on extracting depth from approximate translational motion, intended for practical use in obstacle avoidance on the ALV [DUT88]; extracting depth from stereoscopic motion [BAL88]; and motion analysis that is simplified by assumptions of constant motion [PAV88]; and depth computation from grouped geometric structures [WIL88a,WIL88b]. We will discuss each of these briefly below.

## 3.2 Processing Approximate Translational Motion for the ALV

Previous research in motion analysis led us to attempt to deal with a real application subsystem for the CMU NAVLAB [THOR87]. The goal was to detect obstacles in the path of the vehicle at distances beyond the limits of the ERIM range sensor, i.e. at distances beyond 40 feet. Initial results from Bharwani's algorithm implied the possibility of extracting usable depth of obstacles at distances between 40 and 80 feet. By applying an FOE extraction algorithm prior to the depth extraction algorithm, there was an expectation that an effective subsystem could be developed. To accomplish this in actual imaging situations on a moving vehicle has turned out to be far more difficult.

In dynamic imaging situations where the sensor is undergoing primarily translational motion with a relatively small rotational component, it might seem likely that "approximate" trans-

lational motion algorithms can be effective in determining depth. Although translational motion is the dominant form of motion and is approximately constant over a long sequence of frames, there usually are local variations due to irregularities in the road surface (bumps, holes, and undulations), as well as minor rotation of the vehicle as it translates. This is often manifested in changes in the location of the FOE (i.e. effectively it produces a different translational motion), and in rotational motions that must be removed if correct values of depth are to be extracted from the feature displacements. An attempt to correct for these effects via a relatively simple preprocessing algorithm, without utilizing full analysis of the general motion problem, also led to difficulties. The issues and our experimental efforts to deal with what we considered to be the relatively simple problem of approximate translational motion are discussed in [DUT88] in this volume.

The problems led us recently to apply the Anandan and Adiv algorithms for general motion to the sequences of approximate translational motion with significantly improved results; this approach is also reported in [DUT88]. The conclusion is that while the FOE might be approximately extracted, in many real situations general motion analysis must be applied in order to determine depth of points, even when sensor motion is primarily translational with only small amounts of rotation. One obvious hardware solution (at significantly increased cost) is the use of a gyro-stabilized platform so that sensor motion typically will be much closer to the case of pure translational motion. Alternatives to these approaches to motion processing to extract motion parameters and depth are outlined in the next two subsections, and in the perceptual organization section where spatio-temporal grouping is used to derive depth from geometric structures.

### 3.3 Steroscopic Motion Analysis

By carrying out motion analysis with a pair of cameras - stereoscopic motion - the additional constraints can significantly reduce the complexity of the analysis on a theoretical level. Balasubramanyan and Snyder [BAL87a,BAL87b,BAL88] have developed an algorithm to extract the parameters of motion in depth: the single component of translation in depth (i.e. parallel to the line of sight) and the two components of rotation in depth (i.e. rotations that are not around the line of sight). This is achieved by building upon the work of Adiv for segmenting the flow field into rigid independently moving objects [ADI85a], and the formulation of Waxman and Duncan [WAX86] of the ratio of the relative optic flow between a stereo pair of images to the disparity between them as a linear function of the image coordinates. Experimental results are presented for simulated data of general motion of both the sensor and independently moving objects. Work is currently underway to test the effectiveness on real scenes.

### 3.4 Analysis of Constant General Motion

Another way to introduce additional constraints to the problem of general motion analysis in an effort to achieve practical, robust algorithms is via Shariat's formulation: constant but arbitrary general motion of a rigid object [SHA86]. This leads to a set of difference equations across a sequence of images, relating the positions of a feature in the image plane to the motion parameters of the projected point. The solution obtained is a set of 5th order non-linear polynomial equations in the unknown motion parameters, whose solution requires a Gauss-Newton non-linear least-squares method with carefully designed initial guess schemes. Pavlin [PAV88] has derived a closed-form solution for the rigid object trajectory by integrating the differential equations describing the motion of a point on the tracked object. The integrated equations are non-linear only in angular velocity, and are linear in all other motion parameters. These equations allow the use of a simple least-square error minimization criterion in an iterative search for the motion parameters.

## 4 Perceptual Organization (Grouping)

### 4.1 Token-Based Approaches to Motion and Perceptual Organization

The problems cited in Section 3 with respect to the extraction of motion and depth information using traditional optical flow techniques have led us toward the exploration of methods for combining the local flow/displacement fields with larger token-like structures. It is our position that the inherently local measurement of visual motion provided by optical flow is insufficient to meet the varied requirements of dynamic image understanding. The approach we are developing involves computing the correspondence between tokens of arbitrary spatial scale produced by perceptual organization processes. Such tokens often map directly to environmental structure, and descriptions of their movement often correlate more closely with the motion of physical objects than does the local motion information contained in the displacement field. A token match represents more than just a spatial displacement; also explicit in this representation are the time-varying values of those parameters which define the token or which can be extracted from the structure of the token.

In two papers in this volume, Williams and Hanson [WIL88a,WIL88b] describe work in progress toward this goal. The premise of this work is that the structure obtained from perceptual organization processes can be combined with the local motion information contained in the flow field to provide a more robust estimate of motion and depth parameters. The approach can be viewed as augmenting the rather limited use of spatial structure in tradional approaches with the richer descriptive vocabulary of spatial structure provided by the perceptual organizational processes over both space and time. In this sense, the spatially organized structures (such as lines, regions, curves, vertices, intersections, rectangular groups, etc.), which are actively constructed from the image, can be considered to be interest operators of large spatial extent.

In the first paper [WIL88a], a method for computing the temporal correspondence between straight line segments is presented. We consider the two frame case here, but the method is extensible and has been extended to multiple frames. A straight line perceptual organization process, developed by Boldt and Weiss [BOL87,WEI86], is applied to both frames independently to provide straight lines in each frame. A displacement field is also computed from the two frames using the algorithm developed by Anandan [ANA87b,ANA88]. After filtering the straight lines on length and constrast to reduce the line set in both images, the displacment field is used to construct a search area in frame 2 for each line in frame 1. Since a one-to-one correspondence between lines is unlikely, a minimal mapping approach [ULL79] is used to compute the correspondence between the frame 1 and frame 2 line sets; such a mapping is called a minimal bipartite cover. The similarity measure used to compute the cover involves the similarity and spatial separation of the candidate token matches

By computing the connected components of the bipartite graph, the global matching problem is conveniently divided into smaller, individually tractable pieces which reflect the scope of potential interactions; a simple blind search of the subgraphs is used to extract the bipartite cover minimizing the positional and similarity discrepancy metric.

The matching results obtained are quite good. The system has been run repeatedly on successive frames of several multi-frame sequences. In the multi-frame case, a directed acyclic graph is constructed which represents the splitting and merging patterns of line segments over time. Work is in progress to analyze the trajectories of the tokens over time.

In the second paper [WIL88b], a method for computing depth from the line correspondences is described using the temporal change in the length of virtual lines constructed from the intersections of the Boldt lines [BOL87]. We use virtual lines because the length of the original lines is not reliable, although their orientation and lateral displacement is quite precise. This "looming" method is also generalized to areas. The method is generally applicable to structures whose total extent in depth is small compared to the depth of its centroid (that is, for those cases in which perspective projection can be approximated by scaled orthographic projection [THOM87]) and which do not exhibit any independent motion. The technique does not depend on the complete determination of egomotion parameters of the sensor, but it does require the computation of the translation component of the sensor in the direction of motion. An analysis of the sensitivity of the algorithm to errors in the measured variables is planned for the near future; experimental results on real image sequences have shown that the algorithm may be quite robust.

### 4.2 The Perceptual Organization of Image Curves

Most of our work in perceptual organization [BEL86,BOL87,BUR86,DOL86,DOL88,REY84,REY86b,REY87] [RIS87,WEI85,WEI86,WIL88a,WIL88b] has been focussed on rectilinear structures (e.g. straight lines, corners, parallel line pairs, and the like). Unfortunately, not all of the world can be described by straight lines. Consequently, Dolan [DOL88] has been exploring methods for extending the general technique developed by Boldt [BOL87,WEI86] to the simultaneous extraction of curves, straight lines, and corners (including cusps); these are the primitive descriptive elements. The basic operation cycle consists of linking, grouping, and replacement, which takes place at increasing perceptual scales, resulting in a hierarchical scale-space description of these important image events.

The linking stage finds subsets within the set of initial local edge tokens that satisfy the binary constraints of the particular grouping principles employed. The grouping mechanisms perform a detailed geometric analysis on sets of linked tokens whose extent is within the current scale; in Dolan's system, this also entails classification and ranking of the token sequences as one of the basic primitive elements. Replacement mechanisms encode the geometry of a surviving group by substituting a single token for the group. The process then repeats at the next scale.

### 4.3 Extracting Geometric Structure

Reynolds and Beveridge [REY87] have been developing a perceptual grouping system for the extraction of rectilinear structures from an initial set of line primitives obtained using the straight line extraction algorithm developed by Burns, Hanson, and Riseman [BUR86]. The lines are represented as nodes in a graph. The grouping criteria are the geometric relations of spatially proximate collinear, spatially proximate parallel, spatially proximate orthogonal, or any subset of these relations; the relations form the links in the graph.

Line groups are generated using a connected components analysis of the chosen geometric links. Finally, individual geometric structures (e.g. rectangles, collinear lines, paraliel line pairs, corners, etc.) may be identified as subgraphs of the connected components. These techniques have been applied to extraction of objects such as road networks in aerial images.

Object recognition strategies can be represented as relational graphs to be matched to extracted data. The problems associated with fragmentation, as well as merged and missing tokens, makes this a difficult problem. However, multiple representations (such as lines and regions) can be brought together to provide partial redundancy [RIS87]. Thus, current work overlaps issues of constrained graph matching, perceptual organization, and information fusion.

## 5 Image Understanding Architecture

The Image Understanding Architecture being jointly designed and constructed by the University of Massachusetts and Hughes Research Laboratories [WEE84,WEE87,WEE88a,WEE88b] is a multi-level, heterogeneous, associative, parallel machine to support real-time knowledge-based computer vision. The machine consists of three computational levels (the CAAPP, ICAP, and SPA, respectively), roughly corresponding to the low, intermediate, and high-levels of abstraction currently believed necessary for image understanding [HAN87a]. At each level of the IUA, the processing elements are tuned to the computational granularity of the algorithms required at that particular level of abstraction.

The CAAPP (Content Addressable Array Parallel Processor) is a 512x512 square grid array of custom processors designed to perform low-level image processing tasks. The CAAPP is also specifically designed to interact with the ICAP (through a shared memory) in a tightly coupled fashion for both bottom-up and top-down processing. The ICAP (the Intermediate Communications Associative Processor) is designed to manipulate tokens either extracted from the image data at the CAAPP level or constructed from other tokens at the ICAP level. The ICAP is also a square grid (64x64), built from Texas Instruments TMS320-C25 signal processing chips, each with 256K bytes of local memory, and 384K bytes of dual-ported memory for CAAPP/ICAP and ICAP/SPA communication and data storage. The ICAP can operate in either synchronous MIMD or pure MIMD mode.

The SPA (Symbolic Processing Array) is constructed of processors capable of running a LISP-based blackboard system at each node. The SPA processors operate in MIMD mode with intra-level communication through the blackboard and inter-level communication through the dual-ported memory shared with the ICAP processors. Since only a 1/64th prototype is currently being built, the SPA level will consist of a single processor in the Motorola 68020 class; in the full machine the SPA level will consist of 64 or more processors, each capable of running LISP. The detailed architecture of the SPA level of the full machine has not yet been fully defined, but research in progress oriented towards porting the schema system to a commercial shared-memory multiprocessor is expected to provide insight into its structure.

The CAAPP and ICAP levels are controlled by a dedicated Array Control Unit which contains two separate control processors. The Macro-controller is a 68020-based system which supports the software development environment and provides an interface to the programmer. The Micro-controller is a custom processor driven by horizontal microcode and is capable of issuing an instruction to the CAAPP every 100 nanoseconds. Access to the Micro-controller is through a library of CAAPP control subroutines. In this way, the advantages of a high-level program development environment are combined with the speed advantages of the Micro-controller. The ACU is also accessible from the SPA level, providing knowledge-driven control of both the low-level and the intermediate-level processes. The programmer's model for this environment is described in [WEE88b].

### 5.1 The Coterie Network on the CAAPP

The requirements of high-speed, fine-grained bi-directional inter- and intra-level communication and control have led to the development of very general associative processing techniques to support the communication requirements [WEE88a,WEE88b]. Currently, there are four mechanisms for communication between CAAPP cells. One method used the hardware implementation of the associative processing capabilities to accomplish global feedback and rebroadcast. Communication can also take place through the ICAP level via the backing store. The third mechanism uses the traditional S,E,W,N neighborhood network between adjacent CAAPP processors.

The fourth mechanism involves a new and powerful variation on the nearest neighbor mesh called the Coterie Network. The coterie technique allows the CAAPP mesh to be partitioned into independent groups of processors that share a local associative Some/None circuit. The independent groups of processors can then respond to globally broadcast instructions in a locally data-dependent way, which permits the parallelism in the mesh to be more flexibly exploited. For example, each coterie might correspond to a single region in a region segmentation; each region could then be processed independently and in parallel.

The coterie mechanism is implemented through a network of software controlled switches, one switch between each adjacent processor. Opening the switch between two processors effectively isolates them from communicating with each other over the open line. Creating a closed path of open switches creates an island of processors isolated from the remainder of the mesh. Leaving the switches inside the 'island boundary' closed creates an internal communication network for the processors participating in the coterie. Each processor may write or read a bit from the network; when more than one processor writes to the network, the result is the logical OR of the output bits of the processors. The shared mesh is thus functionally equivalent to the global Some/None network, but local to the coterie. Several image processing algorithms which utilize the Coterie Network are discussed in [WEE88b].

### 5.2 Status

The IUA programming environment currently exists in software simulation on an Explorer LISP workstation, augmented with a Texas Instruments Odyssey parallel signal co-processor. Portions of the simulation are also available on VAX and SUN systems.

At the hardware level, final versions of the custom CAAPP chips are currently being fabricated through MOSIS. A CAAPP-CISM-ICAP-ISSM test structure has been successfully breadboarded by Hughes Research Laboratories. The prototype IUA system is scheduled for completion in the Fall of 1988.

## 6 Integrated Vision Benchmark for Parallel Architectures (DARPA IU Benchmark: Round 2)

The most recent attempt at constructing a vision benchmark for parallel architectures emerged from the DARPA IU community in 1986. This benchmark consists of ten prototypical vision tasks: Gaussian convolution, zero-crossing detection and output of border lists, connected components labeling, Hough transform, determining the convex hull, constructing a minimal spanning tree, computing the visibility of vertices in a 3-D model, finding a minimal cost path, and subgraph matching. Each of these tasks were benchmarked individually and the results reported in [ROS87].

While useful information was gained from this exercise, there were significant shortcomings. In particular, it was recognized that the individual benchmarks did not adequately represent the performance of a machine on an integrated vision task, such as knowledge-based image interpretation. In response a group of researchers from the University of Maryland and the University of Massachusetts accepted the responsibility of formulating an integrated benchmark representing a more realistic interpretation scenario that transcends several different representations and forms of processing that are typical of complex vision applications. The goal was to generate a benchmark which would lead to a better understanding of vision architecture requirements and the performance bottlenecks in different classes of machines. A secondary goal was to utilize as many of the algorithms as possible from the first benchmark in order to minimize the coding impact on participants of the second benchmarking task. The benchmark was developed with some input from other members of the DARPA IU community [WEE88c].

The integrated benchmark involves recognizing an approximately specified "2 1/2 D mobile" sculpture composed of rectangles, given images from intensity and range sensors. The test images are designed so that the data from neither sensor by itself is sufficient to solve the task. The sculpture can be thought of as a semi-rigid mobile consisting of suspended rectangles floating in space with spatial relationships that are fixed only up to some tolerance limit. Each rectangle is oriented normal to the Z axis (the viewing direction) and the image is constructed under orthographic projection. In the image, the rectangles comprising the mobile are interspersed with additional rectangles. The additional rectangles may occlude portions of the mobile object, and some of the adjacent rectangles in the scene may have very similar brightnesses and depths.

A set of symbolic models of several mobile sculptures will be provided when the benchmark is distributed. These models are only approximate in that the sizes, orientations, depths, and spatial relationships of the rectangles are constrained within some tolerance bands. The goal is to determine which of the models are present in the images, the degree to which it is visible (matchable), and to update the model with positional data that has been extracted from the images.

In order to ensure comparable benchmarking results, an algorithm-level solution to the interpretation problem is provided, as well as a set of instrumentation guidelines. It is not

intended that this solution be optimal or even that it constitutes a reasonable approach. The goal is to provide a measure of consistency among the solution methods and benchmark results. The solution has been tested on sequential machines by the University of Massachusetts and the University of Maryland in order to remove ambiguity in its statement and to detect as many unexpected problems in its implementation as possible. The instrumentation guidelines are currently being developed and reviewed, and will be distributed at a later date. The benchmark is intended to be widely distributed, and a workshop for comparison of results on a variety of machines will be held at a future date.

# 7 Mobile Vehicle Navigation

The hardware platform for experimentation in mobile robotics at UMass is a Denning Mobile Robotics vehicle with a B&W television camera and UHF transmitters and receivers for uplink and downlink communication to a Gould IP8500 image processing system connected to a Vax 11/750 computer. Plans are underway to utilize a 12-node Sequent multiprocessor to improve the computational effectiveness of our experimentation environment.

Arkin [ARK87a,ARK87b,ARK87c] used this platform to develop AuRA (Autonomous Robot Architecture), which integrates planning, cartographic, perception, motor, and homeostatic systems into a functional robot navigation system. The system is designed to navigate in the hallways and outdoor environment surrounding our building at UMass.

Aura employs a 'meadow' map as its long-term memory; the meadow map is used for global path planning and contains embedded a priori knowledge to guide sensor expectations used for positional updating. A layered short-term memory based on instantiated meadows represents the currently perceived world. A hierarchical path planner produces a global path free of collisions with all modeled obstacles.

Aura extends the idea of schemas, as currently employed in the VISIONS system, to include the mobile robot domain. The schema-based path execution system handles unexpected and dynamic obstacles not present in the robot's world model. This motor-schema based navigation system produces reactive/reflexive behavior in direct response to sensor events. In addition, new techniques in the treatment of robot uncertainty, which expedite sensory processing, were developed. These include the use of a spatial error map with associated growth and reduction techniques.

Several computer vision sensor strategies have been developed for use within Aura. These include a fast line finding algorithm that is a simplified and more efficient version of the Burns straight line extraction algorithm (at the price of robustness) [BUR86,KAI187], a fast simplified region segmentation algorithm based on the VISIONS region segmentation system [BEV87], and a depth from motion algorithm [BHA86]. Aura uses both vision and ultrasonic sensing during path traversal.

We are currently rebuilding Aura to make better use of the information available from the visual sensors and to more completely integrate the full spectrum of image understanding techniques developed in the VISIONS project. In particular, we intend to utilize some of the depth from motion algorithms discussed above [BAL88,DUT88,WIL88a,WIL88b]; and some of the simpler object recogniton strategies of the schema system [DRA87a,DRA87b,HAN87b], including strategies for multisensor information fusion [BEL86,RIS87].

## REFERENCES


[ADI85a] G. Adiv, "Determining 3-D Motion and Structure from Optical Flow Generated by Several Moving Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Volume PAMI-7, July 1985, pp. 384-401.

[ADI85b] G. Adiv, "Interpreting Optical Flow," Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts at Amherst, September 1985.

[ADI85c] G. Adiv, "Inherent Ambiguities in Recovering 3-D Motion and Structure from a Noisy Flow Field," *Proc. of the Computer Vision and Pattern Recognition Conference*, San Francisco, CA, June 1985, pp. 7077.

[ANA84] P. Anandan, "Computing Dense Displacement Fields with Confidence Measures in Scenes Containing Occlusion," *SPIE Intelligent Robots and Compter Vision Conference,* Volume 521, 1984, pp. 184-194; also *DARPA IU Workshop Proceedings,* 1984; and COINS Technical Report 84-32, University of Massachusetts at Amherst, December 1984.

[ANA85a] P. Anandan and R. Weiss, "Introducing a Smoothness Constraint in a Matching Approach for the Computation of Optical Flow Fields," *Proc. of the Third Workshop on Computer Vision: Representation and Control,* October 1985, pp. 186-196, also in *DARPA IU Workshop Proceedings,* 1985.

[ANA85b] P. Anandan, "Motion and Stereopsis," COINS Technical Report 85-52, University of Massachusetts at Amherst, December 1985; also to appear (in Spanish) in *Vision por Computador,* (Carme Torras, ed.), to be published by Alianza Editorial, Spain.

[ANA87a] P. Anandan, "Measuring Visual Motion From Image Sequences", Ph.D. Dissertation, University of Massachusetts at Amherst, January 1987.

[ANA87b] P. Anandan, "A Unified Perspective on Computational Techniques for the Measurement of Visual Motion", *Proc. of the International Conference on Computer Vision*, London, England, June 1987.

[ANA88] P. Anandan, "A Computational Framework and an Algorithm for the Measurement of Visual Motion", to appear in *International Journal on Computer Vision*, 1988.

[ARK87a] R. Arkin, "Motor Schema-Based Navigation for a Mobile Robot: An Approach to Programming by Behavior", *Proc. of the IEEE International Conference on Robotics and Automation*, Raleigh, NC, pp. 264-271, March 1987.

[ARK87b] R. Arkin, "Towards Cosmopolitan Robots: Intelligent Navigation in Extended Man-Made Environments", Ph.D. Thesis, Computer & Information Science Department, University of Massachusetts at Amherst, September 1987.

[ARK87c] R. Arkin, A. Hanson, and E. Riseman, "Visual Strategies for Mobile Robot Navigation", *Proc. of IEEE Computer Society Workshop on Computer Vision*, Miami, FL November, 1987.

[BAL87a] P. Balasubramanyam, "Extraction of Motion In Depth: A First Step in Stereoscopic Motion Interpretation", presented at the American Institute of Aeronautics and Astronautics Computers in Aerospace IV Conference, Wakefield, MA, October 1987, pp. 277-286.

[BAL87b] P. Balasubramanyam, "Computation of Motion-In-Depth Parameters Using Stereoscopic Motion Constraints", *Proc. of IEEE Computer Society Workshop on Vision*, Miami, FL, November 1987.

[BAL88] P. Balasubramanyam and M. Snyder, "Computation of Motion in Depth Parameters: A First Step in Stereoscopic Motion Interpretation", *Proc. of the DARPA Image Understanding Workshop*, Cambridge, MA, April 1988.

[BEL86] R. Belknap, E. Riseman, and A. Hanson, "The Information Fusion Problem and Rule-Based Hypotheses Applied to Complex Aggregations of Image Events", *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Miami, FL, June 1986, pp. 227-234.

[BEV87] J. Ross Beveridge, J. Griffith, R. Kohler, A. Hanson, and E. Riseman, "Segmenting Images Using Localized Histrograms and Region Merging", COINS Technical Report 87-88, University of Massachusetts at Amherst, September 1987.

[BHA85] S. Bharwani, A. Hanson, and E. Riseman, "Refinement of Environmental Depth Maps over Multiple Frames," *Proc. DARPA IU Workshop*, Miami Beach, FL, December 1985.

[BHA86] S. Bharwani, E. Riseman, and A. Hanson, "Refinement of Environmental Depth Maps Over Multiple Frames", *Proc. of the Workshop on Motion: Representation and Analysis*, Charleston, SC, May 1986, pp. 73-80.

[BOL87] M. Boldt and R. Weiss, "Token-Based Extraction of Straight Lines", COINS Technical Report 87-104, University of Massachusetts at Amherst, October 1987.

[BRO88] J. Brolio, J. Ross Beveridge, B. Draper, J. Griffith, G. Reynolds, R. Collins, J. Burrill, L. Williams, A. Hanson, E. Riseman, "Symbolic Data Management for Vision Research", COINS Technical Report, University of Massachusetts at Amherst, 1988.

[BUR88] J.B. Burns and Leslie J. Kitchen, "Rapid Object Recognition From A Large Model-Base Using Prediction Hierarchies", *Proc. DARPA Image Understanding Workshop*, Cambridge, MA, April 1988. Also COINS Technical Report, University of Massachusetts at Amherst, in preparation.

[BUR87a] J.B. Burns, "Recognition in 2D Imges of 3D Objects from Large Model Bases Using Prediction Hierarchies", *Proc. of the International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987, pp. 763-766.

[BUR87b] J.B. Burns and L.J. Kitchen, "An Approach to Recognition in 2D Images of 3D Objects From Large Model Bases", COINS Technical Report 87-85, University of Massachusetts at Amherst, August 1987.

[BUR86] J.B. Burns, A.R. Hanson, and E.M. Riseman, "Extracting Straight Lines," *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*, No. 4, July 1986, 425-455.

[DEM68] A.P. Dempster, "A Generalizaton of Bayesian Inference", *Journal of the Royal Statistical Society*, Series B., Vol. 30, 1968, pp. 205-247.

[DOL88] John Dolan and Richard Weiss, "The Perceptual Organization of Image Curves", COINS Technical Report, University of Massachusetts at Amherst, in preparation.

[DOL86] J. Dolan, G. Reynolds, and L. Kitchen, "Piecewise Circular Description of Image Curves Using Constancy of Grey-Level Curvature", COINS Technical Report 86-33, University of Massachusetts at Amherst, July 1986.

[DRA88] B. Draper, J. Brolio, R. Collins, A. Hanson, and E. Riseman, "Information Fusion by Distributed Subsystems in a Knowledge-Based Vision Architecture", to appear in *Computer Vision and Pattern Recognition*, 1988.

[DRA87a] B. Draper, R. Collins, J. Brolio, J. Griffith, A. Hanson, and E. Riseman, "Tools and Experiments in the Knowledge-Directed Interpretation of Road Scenes", *Proc. of the DARPA Image Understanding Workshop*, Los Angeles, CA, February 1987, pp. 178-193.

[DRA87b] B. Draper, R. Collins, J. Brolio, A. Hanson, and E. Riseman, "The Schema System", COINS Technical Report, University of Massachusetts at Amherst, in preparation.

[DUT88] R. Dutta, R. Manmatha, E. Riseman, and M.A. Snyder, "Issues in Extracting Motion Parameters and Depth from Approximate Translational Motion", *Proc. DARPA Image Understanding Workshop*, Cambridge, MA, April 1988.

[FIS86] M.A. Fischler and R.C. Bolles, "Perceptual Organization and Curve Partitioning", *IEEE PAMI*, 8(1), pp. 100-105, January 1986.

[GLA87a] F. Glazer, "Hierarchical Motion Detection", Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts, Amherst, February 1987.

[GLA87b] F. Glazer, "Hierarchical Gradient-Based Motion Detection", COINS Technical Report 87-77, University of Massachusetts at Amherst, 1987.

[GLA87c] F. Glazer, "Computation of Optic Flow by Multilevel Relaxation", Coins Technical Report 87-64, University of Massachusetts at Amherst, 1987.

[HAN78a] A.R. Hanson and E.M. Riseman (Eds.), *Computer Vision Systems*, New York, Academic Press, 1978.

[HAN78b] A. Hanson and E. Riseman, "Segmentation of Natural Scenes," in *Computer Vision Systems*, (A. Hanson and E. Riseman, Eds.), Academic Press 1978, pp. 129-163.

[HAN78c] A. Hanson, and E. Riseman, "VISIONS: A Computer System for Interpreting Scenes, in *Computer Vision Systems* (A. Hanson and E. Riseman, eds.) pp. 303-333, Academic Press, 1978.

[HAN86] A.R. Hanson and E.M. Riseman, "A Methodology for the Development of General Knowledge-Based Vision Systems," in *Vision, Brain, and Cooperative Computation*, (M. Arbib and A. Hanson, Eds.) 1986, MIT Press, Cambridge, MA.

[HAN87a] A.R. Hanson and E.M. Riseman, "The VISIONS Image Understanding System", in *Advances in Computer Vision*, Chris Brown, (Ed.), Erlbaum Press, 1987. Also COINS Technical Report 86-62, University of Massachusetts at Amherst, December 1986.

[HAN87b] A. Hanson and E. Riseman, "From Image Measurements to Object Hypotheses", COINS Technical Report 87-129, University of Massachusetts at Amherst, December 1987.

[KAH87] P. Kahn, L. Kitchen, and E. Riseman, "Real-Time Feature Extraction: a Fast Line Finder for Vision-Guided-Robot Navigation", COINS Technical Report 87-57, University of Massachusetts at Amherst, July 1987.

[KOH87] C.A. Kohl, A.R. Hanson, and E.M. Riseman, "A Goal-Directed Intermediate Level Executive for Image Interpretation", *Proc. of the International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987, pp. 811-814.

[KOH88] C.A. Kohl, "Goal-Directed Control for Computer Vision", Ph.D. Dissertation, Computer & Information Science, University of Massachusetts at Amherst, February 1988.

[LAW84] D.T. Lawton, "Processing Dynamic Image Sequences from a Moving Sensor," Ph.D. Dissertation (COINS Technical Report 84-05), Computer and Information Science Department, University of Massachusetts, 1984.

[LAW83] D.T. Lawton, "Processing Translational Motion Sequences", *Computer Graphics and Image Processing*, Vol. 22, pp. 116-144, 1983.

[LEH87] N. Lehrer, G. Reynolds, and J. Griffith, "A Method for Initial Hypothesis Formation in Image Understanding", COINS Technical Report, University of Massachusetts at Amherst, in preparation, 1987.

[LOW85] D.G. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, 1985.

[PAV85] I. Pavlin, A. Hanson, and E. Riseman, "Analysis of an Algorithm for Detection of Translational Motion," *Proc. DARPA IU Workshop*, Miami Beach, FL, December 1985.

[PAV86] I. Pavlin, E. Riseman, and A. Hanson, "Translational Motion Algorithm With Global Feature Constraints", COINS Technical Report 86-58, University of Massachusetts at Amherst, December 1986.

[PAV88] I. Pavlin, "Motion From a Sequence of Images", *Proc. of the DARPA Image Understanding Workshop*, Cambridge, MA, April 1988.

[REY86a] G. Reynolds, D. Strahman, N. Lehrer, L. Kitchen, "Plausible Reasoning and the Theory of Evidence", COINS Technical Report 86-11, University of Massachusetts at Amherst, April 1986.

[REY86b] G. Reynolds, J. Ross Beveridge, "Geometric Line Organization Using Spatial Relations and a Connected Components Algorithm", COINS Technical Report, University of Massachusetts at Amherst, in preparation, 1986.

[REY84] G. Reynolds, N.Irwin, A. Hanson and E. Riseman, "Hierarchical Knowledge- Directed Object Extraction Using a Combined Region and Line Representation," *Proc. of the Workshop on Computer Vision: Representation and Control*, Annapolis, Maryland, April 30 - May 2, 1984, pp. 238-247.

[REY85] G. Reynolds, D. Strahman, N. Lehrer, "Converting Feature Values to Evidence," *Proc. DARPA IU Workshop*, Miami Beach, FL, 1985.

[REY87] G. Reynolds and J.R. Beveridge, "Searching For Geometric Structure in Images of Natural Scenes", COINS Technical Report 87-03, University of Massachusetts at Amherst, January 1987.

[RIS87] E. Riseman, A. Hanson, and R. Belknap, "The Information Fusion Problem: Forming Token Aggregations Across Multiple Representations", COINS Technical Report 87-48, University of Massachusetts at Amherst, 1987.

[RIS84] E.M. Riseman and A.R. Hanson, "A Methodology for the Development of General Knowledge-Based Vision Systems," *IEEE Proc. of the Workshop on Computer Vision: Representation and Control*, 1984, pp. 159-170.

[ROS87] A. Rosenfeld, "A Report on the DARPA Image Understanding Architectures Workshop", *Proc. of the DARPA Image Understanding Workshop*, Los Angeles, CA, February 1987, pp. 298-302.

[SHA76] G. Shafer, "A Mathematical Theory of Evidence", Princeton University Press, 1976.

[SHA86] H. Shariat, "The Motion Problem: How To Use More Than Two Frames", Ph.D. Dissertation, University of Southern California, Los Angeles. Also Technical Report IRIS 202, Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles, CA, October 1986.

[SNY86] M. A. Snyder, "The Accuracy of 3D Parameters in Correspondence-Based Techniques," *Proc. of the Workshop on Motion: Representation and Analysis,* Charleston, S.C., May 1986; also COINS Technical Report 86-28, University of Masschusetts at Amherst, July 1986.

[SNY88] M.A. Snyder, "The Precision of 3D Parameters in Correspondence-Based Techniques: The Case of Uniform Translational Motion in a Rigid Environment", to appear in *IEEE PAMI*, 1988.

[THOR87] C. Thorpe, S. Shafer, and T Kanade, "Vision and Navigation for the Carnegie Mellon Navlab", *Proc. of the DARPA Image Understanding Workshop*, Los Angeles, CA, February 1987, pp. 143-152.

[THOM87] D. Thompson and J. Mundy, "Three Dimensional Model Matching From an Unconstrained Viewpoint, *Proc. of the IEEE Conference on Robotics and Automation*, Raleigh, NC, 1987.

[ULL79] S. Ullman, "The Interpretation of Visual Motion", MIT Press, Cambridge, MA, 1979.

[WAX86] A.M. Waxman and J.H. Duncan, "Binocular Image Flows: Steps Toward Stereo-Motion Fusion", *IEEE PAMI*, Vol. PAMI-8, November 1986, pp. 715-729.

[WEE84] C. Weems, "Image Processing on a Content Addressable Array Parallel Processsor", Ph.D. Dissertation and COINS Technical Report 84-14, University of Massaschusetts at Amherst, September 1984.

[WEE87] C. Weems, S. Levitan, A. Hanson, and E. Riseman, "The Image Understanding Architecture Workshop", *Proc. of the DARPA Image Understanding Workshop*, Los Angeles, CA, February 1987, pp. 483-498.

[WEE88a] C. Weems, S. Levitan, A. Hanson, E. Riseman, G. Nash, and D. Shu, "The Image Understanding Architecture", to appear in The International Journal on Computer Vision, 1988. Also Technical Report 87-76, Computer & Information Science, University of Massachusetts at Amherst, August 1987.

[WEE88b] Charles Weems, "Some Sample Algorithms for the Image Understanding Architecture", *Proc. DARPA Image Understanding Workshop*, Cambridge, MA, April 1988.

[WEE88c] C. Weems, E. Riseman, A. Hanson, and A. Rosenfeld, "An Integrated Image Understanding Benchmark: Recognition of a 2 1/2 D 'Mobile'", *Proc. DARPA Image Understanding Workshop*, Cambridge, MA, April 1988.

[WEI86] R. Weiss and M. Boldt, "Geometric Grouping Applied to Straight Lines", *Proceedings on the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Miami, Florida, June, 1986, pp. 489-495.

[WEI85] R. Weiss, A. Hanson, and E. Riseman, "Geometric Grouping of Straight Lines," *Proc. 1985, DARPA IU Workshop,* Miami Beach, FL, 1985

[WES82] L. Wesley and A. Hanson, "The Use of Evidential-Based Model for Representing Knowledge and Reasoning About Images in the VISIONS System", *Proc. Workshop on Computer Vision*, Rindge, NH, August 1982.

[WES83] L. Wesley, "Reasoning About Control: The Investigation of an Evidential Approach", *Proc. 8th IJCAI*, Karlsruhe, West Germany, August 1983, pp. 203-210.

[WES86] L. Wesley, "Evidential Knowledge-Based Computer Vision", *Optical Engineering*, Vol. 25, No. 3, March 1986, pp. 363-379.

[WES88] L. Wesley, "Evidential-Based Control in Knowledge-Based Systems", Ph.D. Dissertation, Computer & Information Science, University of Massachusetts at Amherst, February 1988.

[WEY86] T.E. Weymouth, "Using Object Descriptions in a Schema Network For Machine Vision," Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts at Amherst. Also COINS Technical Report 86-24, University of Massachusetts at Amherst, 1986.

[WIL86] Lance R. Williams and P. Anandan, "A Coarse-to-Fine Control Strategy for Stereo and Motion on a Mesh-Connected Computer", COINS Technical Report 86-19, University of Massachusetts at Amherst, May 1986.

[WIL88a] L. Williams and A. Hanson, "Depth From Looming Structure", *Proc. of the DARPA Image Understanding Workshop*, Cambridge, MA, April 1988

[WIL88b] L. Williams and A. Hanson, "Translating Optical Flow Into Token Matches", *Proc. of the DARPA Image Understanding Workshop*, Cambridge, MA, April 1988.

[WIT83] A.P. Witkin and J.M. Tenenbaum, "What Is Perceptual Organization For?", *IJCAI*, 1983, pp. 1023-1026.

# Progress in Image Understanding at the University of Rochester

Christopher M. Brown
Computer Science Department
University of Rochester
Rochester, NY 14627

## 1. Active Vision

The active vision paradigm is becoming an increasingly popular one in the image understanding community. Rochester's work in this area has been mostly theoretical and tool-building during 1987, and we expect implementations to develop reasonably rapidly. Pilot projects such as ROVER [Coombs and Marsh 1987; Coombs 1987] are being extended to incorporate new hardware and software tools as they are commissioned.

Eye movements and a foveated retina are features of animal visual systems that are beginning to be seriously considered in computer vision. The computational advantages provided by eye and head movements as well as foveation are being considered at Rochester. Foveation provides mathematical simplifications in vision constraint equations, and is clearly of use in high-resolution examination and focus of attention. Foveation maintained over time leads to tracking, which has its own advantages in computing egomotion [Bandopadhay and Ballard 1987]. Several advantages of active (i.e., multiframe) vision and the tracking of objects in the visual field are given in [Aloimonos et al. 1987].

Eye movements simplify the eye's information-gathering function in several ways. They provide constraints that simplify the low-level computation of visual features, such as optic flow and relative depth. Eye movements and a learning algorithm can solve the visuo-motor calibration problem that geometrically relates the observer and its effectors to the environment. Foveation can constrain pre-attentive perception, thus alleviating the indexing problem--which object accounts for which salient features. Visual problem-solving is known to correlate with eye movements.

The primate fovea is a feature not shared by most computer vision systems. Despite making three or four saccades per second and the 150 millisecond saccadic latency (which may have to do with the computation of where to look next), the human system spends most of its time fixating a point of visual space. The fovea and the ability to fixate objects upon it yield several computational advantages as well as the higher-level concept of focus of attention. In our current work, foveation plays a key role.

Tom Olson is working on a two-stage model of motion processing inspired by theories of primate motion processes. A retinotopic stage that computes simple functions of contrast distribution is followed by a non-retinotopic stage concerned with segments and their trajectories. This model is being formalized in a connectionist net representation. It is meant to complement the work of Nigel Goddard, who is exploring the use of motion information in object recognition.

One example of the benefits of tracking is computing relative depth from parallax change that is under control of the observer. This computation of depth from head motions is currently being implemented in our lab. In [Bandopadhay and Ballard 1987; Bandopadhay 1986] the foveal constraint was carried further to estimate egomotion (or rigid body motion). Knowing the depth from parallax simplifies those calculations. Further, we can select the foveal frame and use the known angular tracking velocities from the pursuit system in the solution. Finally, we can use constraints arising from programmed eye motions and physical construction of the eye (e.g., that rotation around the optic axis is small). The resulting system is much more highly constrained than when optic flow is given to a passive observer ignorant of the conditions under which the flow arose, and correspondingly more facts can be deduced about the environment.

The computational requirements for cooperation between sensing and motion are quite severe, and it has only been recently that technology has allowed off-the-shelf components to be assembled into a serious attempt at real-time sensorimotor cooperation of the sort that is required. Not only must the feedback and feedforward between sensors and effectors be fast, but the style of control must be developed. Modes of control must cooperate intimately and be dynamically configurable, as when the observer uses optic flow to detect an object to track and then, when tracking, ignores optic flow from the background.

Work on the learning of complex sensorimotor skills in connectionist networks uses the idea of teaching the network simple skills initially, which can be combined to attack more complex tasks. The resulting network has an architecture that reflects the hierarchy of tasks and their interconnections. Teaching the system with reinforcement techniques can proceed in parallel, with each learning network operating independently to find maximally rewarding actions [Feldman et al. 1988a; 1988b; Lynne 1987b].

## 2. Object Recognition

There are two active projects in object recognition from a large database. Michael Swain is working on how to structure a database for fast recognition from a large set of models. In particular, he is looking at how a decision tree can be used for such a task. The use of decision trees in the object recognition domain is heavily associated with the pattern recognition paradigm in which each feature may be represented by an axis in a feature space, and in which an object is a point in that space. Swain's domain does not have the independent features of pattern recognition; instead local features are composed and represented by a labeled graph. The aim of his work is to show that a decision tree is a viable approach in this richer domain.

To demonstrate and test the decision tree approach he has designed a system that recognizes polyhedra in crowded scenes. A decision tree with near-optimal expected runtime is constructed, based on the probability distribution of the objects and their viewpoints, and the probabilities of the errors which may occur in the input. A minimum information entropy approach is used as a heuristic for building a near-optimal tree.

The root of the decision tree is the information from a single vertex. The state of the recognition system, represented by a location in the decision tree, moves towards the leaves as it makes tests on the image data structure. The image structure is a graph representing the line segments and vertices in the image labeled with the lengths of the line segments and the angles at the vertices. Which test to make is determined by the state of the recognition system, which is in turn determined by the results of past tests.

Paul Cooper pursued two lines of work in object recognition in 1987, first investigating uncertainty and inexactness in parallel structure recognition, and second investigating the role of domain dependence. The correctness of the connectionist structure recognizer described in [Cooper and Hollbach 1987] was proved using techniques from this research, and a complexity analysis provided [Cooper 1987b]. This work has led to collaboration with Mike Swain on a parallel implementation of well-known constraint algorithms. This last work will be reported at a later date. The role of domain dependence can be modeled by providing precise semantics for "near match," and work is progressing on subgraph isomorphism with constrained types of graphs.

Related to the object recognition work are theoretical and practical results on principal views. Watts [1987] developed theory and a practical algorithm using plane-sweeping techniques for enumerating explicitly the principal views of convex planar polyhedra. Brown developed code that approximately enumerates principal views of blob-and-stick constructions. Watts is extending her work to non-convex polyhedra. Principal views can be of use in optimal recognition algorithms such as that of Swain, and are used in view catalogs like that of Cooper and Hollbach.

## 3. Multi-Modal Segmentation with Markov Random Fields

During the last year David Sher continued his work on probabilistic techniques and foundations in computer vision. He developed the role of likelihoods in designing optimal low-level feature-finders. In work this year he combined optimal low-level operators developed using theory from [Sher 1987a; 1987f] with different spatial support. The combination proceeded according to the theory developed in [Sher 1987b], the key idea of which is to use maximum entropy to control the combination. The result was an operator that scales itself to an appropriate spatial support automatically. Further work in this area led to a probabilistic analysis of template matching that uses the same theory developed for edge detection. The results of this work are that template matching can be characterized with likelihoods that are appropriate for evidence combination using Bayesian techinques. Thresholds can then be set for the output and input of the template matcher. Template weights can be derived from knowledge of the error rates of the input detector. Last, the effects of context, namely occluding and neighboring objects, can be modeled with a Markov random field. This work, applied to ideal, synthetic, and real images, is described in Sher [1987c; 1987d; 1987e].

Continuing the probabilistic approach to vision and moving from low-level operators to the problem of segmentation, Paul Chou has combined information from various information sources [Chou and Brown 1987a; 1987b]. In this approach, observable evidence from disparate sources is coherently and consistently combined through a hierarchically structured knowledge tree. Prior knowledge of spatial interactions is modeled with Markov random fields. A posteriori probabilities of segmentations are maintained incrementally with Bayesian probability theory. Using a novel deterministic optimization algorithm [Chou

and Raman 1987; Chou, Brown and Raman 1987], good results and predictable behavior were obtained at once. Briefly, the Highest Confidence First (HCF) algorithm does a gradient descent in a label space that is augmented by an extra label, the uncommitted label. The idea is that in a cooperative network, a node that knows less about what to do (due to weak evidence or uncommitted neighbors) should delay its decision until those with better knowledge have committed. Further, an early decision should be altered if strong opposition from other nodes is later encountered.

Current work is concentrating on applying the HCF algorithm to evidence combination with labels having continuous values (say depth) as well as discrete (say labels "edge" and "no edge"). In other extensions, we are combining "sparse" data with denser data, and data that arrives distributed over time and space. We are also considering how to implement HCF-like algorithms in parallel (pure HCF involves sorting elements by their confidence) and implementing evidence combination on the Butterfly® Parallel Processor.

## 4. High-Level Issues in Vision

Jerry Feldman and his students are continuing their study of higher-level processes and connectionist models. Not only are concept learning, concept representation, and inference being investigated under other funding, but new results on motion vision should be forthcoming soon. In the meantime, fundamental work that supports the work appeared in [Feldman 1988a; 1987a; 1987b; 1987c; 1988b; 1988c; 1988d]. Connectionist learning theory is under investigation by Lynne as mentioned above, and also by Ballard and his students [Simard et al. 1987].

Traditional robot problem solving must be integrated with a real-time approach if intelligent automated agents are to deal with the real world. Recent work has addressed some issues of performance as well as competence of planners [Hartman 1987; Hartman and Tenenberg 1987a; 1987b].

## 5. Hardware and Systems Support for Parallel Vision

The Computer Vision and Robotics Laboratory facilities have been enhanced by several significant hardware additions to integrate our heterogeneous hardware into a configuration designed to support parallel, pipelined, and real-time image analysis and sensor control.

Our 16-processor Butterfly® Parallel Processor (BPP) was upgraded with the floating point platform kit, to provide much faster scientific computation. Our 3-node BPP was upgraded to a Butterfly Plus®. A VME bus connection between the 16-node BPP and the fast Sun-3 vision computer was established, which also connects both computers to a MaxVideo® pipelined parallel image analysis device. Input is provided by our binocular, three degree-of-freedom "robot head," built as a joint project with the University's Mechanical Engineering Department. One motor controls pitch or altitude (a "nodding" motion of the eye platform), and separate motors control each camera's yaw or azimuth (a "verging" action of the individual cameras). The motors have a resolution of 2,500 positions per revolution and a maximum speed of 150 degrees per second. The controllers allow sophisticated velocity and position commands and data readback. The MaxVideo real-time pipelined image processor not only digitizes but has several boards each containing a computationally intensive vision operation (such as an 8×8 convolution, performed at full video rate, or histogramming and feature location extraction). One board contains an entire signal-processing computer. The speed of the pipeline device is needed to support implementations of active vision theories. Also needed is a flexible head positioner, and we have taken delivery of a Puma 761 robot arm which we shall use initially for controllable head positioning and movements, but which will find use later in further robotic research.

Largely under DARPA Strategic Computing (SC) Support, we have been developing software environments for parallel vision program development [e.g., Olson, Bukys and Brown 1987]. The computer systems aspect of this work is reported in Rochester's Butterfly Project Report series, and in other reports and papers (see references under Crowl, Dibble et al., Ellis and Olson, Finkel et al., Fowler et al., Friedberg, Friedberg and Peterson, LeBlanc, LeBlanc and Jain, LeBlanc and Mellor-Crummey, Mellor-Crummey, Mellor-Crummey et al., Scott, Scott and Cox, Scott and Finkel, Scott and LeBlanc, and Scott and Yap). In many cases, code has been distributed to BBN-ACI in Cambridge, MA, for further redissemination.

The connectionist simulator, distributed now to more than 200 sites, has an interactive graphics interface that increases its usefulness considerably [Lynne 1987a].

## 6. Vision Theory

Work on fundamental vision algorithms has progressed on several fronts. The active vision paradigm inspired theoretical work on motion, projection, shape from patterns, tracking, eye movements, mathematical combination of multiple types of image information, the Hough transform, and texture modeling [Aloimonos et al. 1987; Aloimonos and Swain 1987a; 1987b; Bandopadhay 1986; Bandopadhay and Ballard 1987; Ballard et al. 1987; Brown 1987a; 1987b; 1988b; Brown and Aloimonos 1988; Brown et al. 1987; Brown, Hinkelman and Jain

1987]. An efficient algorithm, realizable in hardware, for image smoothing, was developed [Basu and Brown 1987]. Work on stereo vision [Cooper, Friedman and Wood 1987; Cooper 1987a] has continued and culminated in an algorithm that uses information about relative feature positions to constrain strongly the possible matches in a dynamic programming approach. The constraints are powerful enough that the dynamic programming phase has little to do, and the system's implementation gives very robust results on natural scenes with smooth surfaces, blob-and-stick discontinuous surfaces, and scenes illuminated with structured light.

**References**


Aloimonos, J., A. Bandopadhay, and I. Weiss, "Active vision," *Proc., 1st Int'l. Conf. on Computer Vision*, pp. 35-54, London, England, June 1987.

Aloimonos, J. and M.J. Swain, "Paraperspective projection: Between orthography and perspective," TR CAR-TR-320, Center for Automation Research, U. Maryland, May 1987a.

Aloimonos, J. and M.J. Swain, "Shape from patterns: Regularization," TR CAR-TR-283, Center for Automation Research, U. Maryland, April 1987b.

Ballard, D.H., C.M. Brown, D.J. Coombs, and B.D. Marsh, "Eye movements and computer vision," *1987-88 Computer Science and Engg. Research Review*, Comp. Sci. Dept., U. Rochester, Oct. 1987.

Bandopadhay, A., "A computational study of rigid motion perception," Ph.D. Thesis and TR 221, Computer Science Dept., U. Rochester, *December 1986*.

Bandopadhay, A. and D.H. Ballard, "Active navigation: Egomotion perception by the tracking observer," submitted, *Computational Intelligence*, 1987.

Basu, A. and C.M. Brown, "Algorithms and hardware for efficient image smoothing," *CVGIP 40*, 2, pp. 131-46, November 1987.

Brown, C.M. (Ed). *Advances in Computer Vision* (Vols. I and II). Hillsdale, NJ: Lawrence Erlbaum Assoc., Pub., 1988a.

Brown, C.M., "The Hough transform," in S. Shapiro (Ed). *Encyclopedia of Artificial Intelligence*. New York: John Wiley and Sons, Inc., 1987a.

Brown, C.M., "Parallel vision with the Butterfly® computer," invited paper, *3rd Int'l. Conf. on Supercomputing*, to appear, April 1988b.

Brown, C.M., "A space-efficient Hough transform implementation for object location," in E. Wegman (Ed). *Statistical Image Processing and Graphics*. Marcel-Dekker, 1987b.

Brown, C.M. and J. Aloimonos, "Robust computation of intrinsic images from multiple cues," in Brown, C.M. (Ed). *Advances in Computer Vision* (Vol. 1). Hillsdale, NJ: Lawrence Erlbaum Assoc., Pub., 1988.

Brown, C.M., J. Aloimonos, M. Swain, P. Chou, and A. Basu, "Texture, contour, shape, and motion," *Pattern Recog. Letters 5*, 2, pp. 151-68, 1987.

Brown, C.M., E. Hinkelman, and S. Jain, "Parameterization of mottle textures," TR 217, Comp. Sci. Dept., U. Rochester, June 1987.

Chou, P.B. and C.M. Brown, "Multi-modal segmentation using Markov random fields," *Proc., DARPA Image Understanding Workshop*, pp. 663-70, February 1987a.

Chou, P.B. and C.M. Brown, "Probabilistic information fusion for multi-modal image segmentation," *Proc., Int'l. Joint Conf on Artificial Intelligence*, Milan, Italy, August 1987b.

Chou, P.B., C.M. Brown, and R. Raman, "A confidence-based approach to the labeling problem," *Proc., IEEE Workshop on Computer Vision*, November 1987

Chou, P.B. and R. Raman, "On relaxation algorithms based on Markov random fields," TR 212, Computer Science Dept., U. Rochester, July 1987.

Coombs, D.J., "Rover programmer's guide," Internal Publication, Computer Science Dept., U. Rochester, May 1987

Coombs, D.J. and B.D. Marsh, "ROVER: A prototype active vision system," TR 219, Computer Sci. Dept., U. Rochester, Aug. 1987.

Cooper, P.R., "Order and structure in stereo correspondence," TR 216, Computer Science Dept., U. Rochester, June 1987; submitted, *Int'l. J. of Computer Vision*, 1987a.

Cooper, P.R., "Structure recognition by connectionist relaxation," submitted, *Canadian Artificial Intelligence Conf.*, Nov. 1987b.

Cooper, P.R., D.E. Friedman, and S.A. Wood, "The automatic generation of digital terrain models from satellite images by stereo," *Acta Astronautica 15*, 3, pp. 171-180, March 1987.

Cooper, P.R. and S.C. Hollbach, "Parallel recognition of objects comprised of pure structure," *Proc., DARPA Image Understanding Workshop*, pp. 381-391, February 1987.

Crowl, L. ..., "An interface between object-oriented systems," TR 211, Computer Science Dept., U. Rochester, April 1987.

Dibble, P.C., M.L. Scott, and C.S. Ellis, "Bridge: A high-performance file system for parallel processors," submitted, *8th Int'l. Conf. on Distributed Computing Systems*, November 1987.

Ellis, C.S. and T.J. Olson, "Parallel first fit memory allocation," *Proc., IEEE Int'l. Conf. on Parallel Proc'g.*, pp. 502-11, Aug. 1987.

Feldman, J.A., "Computational constraints on higher neural representations," in E. Schwartz (Ed). *Proc., System Development Foundation Symp. on Computational Neuroscience*. Cambridge, MA: Bradford Books/MIT Press, to appear, April 1988a.

Feldman, J.A., "Connectionist representation of concepts," in D.L. Waltz and J.A. Feldman (Eds). *Connectionist Models and their Applications*. Norwood, NJ: Ablex Publishing Company, 1987a.

Feldman, J.A., "Energy methods in connectionist modelling," in P.A. Devijver and J. Kittler (Eds). *Pattern Recognition, Theory and Applications (Vol. F30)*. Berlin: Springer Verlag, 1987b.

Feldman, J.A., "A functional model of vision and space," in M. Arbib and A. Hanson (Eds). *Vision, Brain and Cooperative Computation*. Cambridge, MA: MIT Press/Bradford Books, 1987c.

Feldman, J.A., "Massively parallel computational models," in M.L. Commons et al. (Eds). *Quantitative Analyses of Behavior, Vol. 8: Pattern Recognition and Concept Formation in Animals, People, and Machines*. Hillsdale, NJ: Lawrence Erlbaum Associates, Pub., to appear, 1988b.

Feldman, J.A., "Neural representation of conceptual knowledge," in N. Sharkey (Ed). *Review of Cognitive Science*. Ablex Pub. Corp., to appear, 1988c.

Feldman, J.A., "Structured neural networks in nature and in computer science," in R. Eckmiller and C. von der Malsburg (Eds). *Neural Computers*. Berlin: Springer Verlag, to appear, 1988d.

Feldman, J.A. and C.M. Brown, "Recent progress of the Rochester image understanding project," *Proc., DARPA Image Understanding Workshop*, pp. 65-70, February 1987

Feldman, J.A., M.A. Fanty, and N. Goddard, "Computing with structured neural networks," to appear, *IEEE Computer*, 1988a.

Feldman, J.A., M.A. Fanty, N. Goddard, and K. Lynne, "Computing with structured connectionist networks," TR 213, Comp. Sci. Dept., U. Rochester, April 1987; to appear, *CACM*, 1988b.

Finkel, R.A., M.L. Scott, Y. Artsy, and H.-Y. Chang, "Experience with Charlotte: Simplicity and function in a distributed operating system," *IEEE Trans. on Software Engg.* (Special Issue: Design Principles for Experimental Distrib'd. Systems), to appear, 1988.

Fowler, R.J., T.J. LeBlanc, and J.M. Mellor-Crummey, "A toolkit approach to parallel program debugging and analysis," extended abstract, submitted, *ACM Workshop on Parallel and Distributed Debugging* (May 1988), December 1987.

Friedberg, S.A., "Hierarchical process composition," *Proc., Carnegie Mellon U. Software Engg. Inst. 2nd Workshop on Large-Grained Parallelism*, pp. 28-30, Hidden Valley, PA, October 1987.

Friedberg, S.A., "Transparent reconfiguration requires a third party connect," TR 220 (revised), Comp. Sci. Dept., U. Rochester, Nov. 1987; sub'd., *Distributed Computing Systems Conf*, Oct. 1987.

Friedberg, S.A. and G.L. Peterson, "An efficient solution to the mutual exclusion problems using weak semaphores," *Information Processing Letters 25*, 5, pp. 343-347, 10 July 1987.

Hartman, L., "On practical robot problem solving," *Proc., IEEE International Symposium on Intelligent Control*, pp. 103-108, Philadelphia, PA, January 1987.

Hartman, L. and J. Tenenberg, "Naive physics and the control of inference," TR 207, Computer Science Dept., U. Rochester, March 1987; *Proc., 2nd U. Buffalo Graduate Conf. on Computer Science*, TR 87-4, Computer Science Dept., SUNY Buffalo, March 1987a.

Hartman, L., and J. Tenenberg, "Performance in practical problem solving," *Proc., Int'l. Joint Conf. on Artificial Intelligence*, pp. 535-540, August 1987; TR 203, C mputer Science Dept., U. Rochester, June 1987b.

LeBlanc, T.J., "Problem decomposition and communication tradeoffs in a shared-memory multiprocessor, *Proc., Workshop on Numerical Algorithms for Parallel Computer Architectures*, Springer-Verlag, 1987a.

LeBlanc, T.J., "Shared memory versus message-passing in a tightly-coupled multiprocessor: A case study," BPR 3 (revised), Computer Science Dept., U. Rochester, February 1987b.

LeBlanc, T.J., "Structured message passing on a shared-memory multiprocessor," *Proc., 21st Annual Hawaiian Int'l. Conf. on System Sciences*, January 1988.

LeBlanc, T.J. and S. Jain, "Crowd control: Coordinating processes in parallel," *Proc., Int'l. Conf. on Parallel Processing*, pp. 81-84, August 1987.

LeBlanc, T.J. and J.M. Mellor-Crummey, "Debugging parallel programs with instant replay," *IEEE Trans. on Computers C-36* (Special Issue on Parallel and Distributed Computation), 4, pp. 471-482, April 1987.

Lynne, K.J., "Competitive reinforcement learning," submitted, *IEEE Machine Learning Conference* (June 1988), Dec. 1987b.

Lynne, K.J., "Graphics interface to Rochester Connectionist Simulator," Internal Publication, Computer Science Dept., U. Rochester, May 1987a.

Mellor-Crummey, J.M., "Concurrent queues: Practical fetch-and-$\Phi$ algorithms," TR 229, Computer Science Dept., U. Rochester, October 1987b.

Mellor-Crummey, J.M., "Experiences with the BBN Butterfly," invited paper, to appear, *Proc., COMPCON Spring '88*, San Francisco, CA, February 1988.

Mellor-Crummey, J.M., "Parallel program debugging with partial orders," *Proc., 2nd U. Buffalo Graduate Conf. on Computer Science*, TR 87-4, Comp. Sci. Dept., SUNY Buffalo, March 1987a.

Mellor-Crummey, J.M. and T.J. LeBlanc, "Instrumentation for distributed systems," Position Paper, *Workshop on Instrumentation for Distributed Computing Systems*, January 1987.

Mellor-Crummey, J.M., T.J. LeBlanc, L.A. Crowl, N.M. Gafter, and P.C. Dibble, "Elmwood---An object-oriented multiprocessor operating system," TR 226 and BPR 20, Computer Science Dept., U. Rochester, September 1987; *1987-88 Computer Science and Engineering Research Review*, Computer Science Dept., U. Rochester, October 1987; submitted, *Software--Practice and Experience*, September 1987.

Olson, T.J., L. Bukys, and C.M. Brown, "Low-level image analysis on an MIMD architecture," *Proc., First IEEE Int'l. Conf. on Computer Vision*, pp. 468-475, London, England, June 1987.

Scott, M.L., "Language support for loosely-coupled distributed programs," *IEEE Trans. on Software Engineering SE-13* (Special Issue on Distributed Systems), 1, pp. 88-103, January 1987.

Scott, M.L. and A.L. Cox, "An empirical study of message-passing overhead," *Proc., 7th Int'l. Conf. on Distributed Computing Systems*, pp. 536-543, Berlin, West Germany, September 1987.

Scott, M.L. and R.A. Finkel, "A simple mechanism for type security across compilation units," *IEEE Trans. on Software Engineering*, to appear, 1988.

Scott, M.L. and T.J. LeBlanc, "Psyche: A general-purpose operating system for shared-memory multiprocessors," BPR 19 and TR 223, Computer Science Dept., U. Rochester, August 1987.

Scott, M.L. and S.-K. Yap, "A grammar-based approach to automatic dialogue generation," to appear, *Proc., Computer-Human Interaction Conf.*, May 1988.

Sher, D., "Advanced likelihood generators for boundary detection," TR 197, Computer Science Dept., U. Rochester, January 1987a.

Sher, D., "Evidence combination using likelihood generators," TR 192, Computer Science Dept., U. Rochester, January 1987; *Proc., DARPA Image Understanding Workshop*, pp. 655-62, Feb. 1987b.

Sher, D., "Generating robust operators from specialized ones," *Proc., IEEE Workshop on Computer Vision*, November 1987c.

Sher, D., "A probabilistic analysis of template matching," Working Paper, Comp. Sci. Dept., U. Rochester, Summer 1987d.

Sher, D., "A probabilistic approach to low-level vision," Ph.D. Thesis and TR 232, Computer Sci. Dept., U. Rochester, Oct. 1987e.

Sher, D., "Tunable facet model likelihood generators for boundary pixel detection," *Proc., IEEE Workshop on Computer Vision*, November 1987f.

Simard, P.Y., M. Ottaway, and D.H. Ballard, "Analysis of recurrent backpropagation," submitted, *Snowbird Conf. on Neural Networks* (April 1988), December 1987.

Waltz, D.L. and J.A. Feldman (Eds). *Connectionist Models and their Applications*. Norwood, NJ: Ablex Publishing Co., 1987.

Watts, N., "Calculating the principal views of a polyhedron," TR 234, Computer Science Dept., U. Rochester, December 1987.

# Image Understanding and Robotics Research at Columbia University

John R. Kender[1]
Peter K. Allen
Terrance E. Boult
Hussein A. H. Ibrahim

Department of Computer Science
Columbia University, New York, NY 10027

## 0 Introduction

The research investigations of the Vision/Robotics Laboratory at Columbia University reflect the diversity of interests of its four faculty members, two staff programmers, and 15 Ph.D. students. Several of the projects involve either a visiting computer science post-doc, other faculty members in the department or the university, or researchers at AT&T Bell Laboratories or Philips Laboratories. We list below a summary of our interest and results, together with the principal researchers associated with them. Since it is difficult to separate those aspects of robotic research that are purely visual from those that are vision-like (for example, tactile sensing) or vision-related (for example, integrated vision-robotic systems), we have listed all robotic research that is not purely manipulative.

### 0.1 Low-level Vision

#### 0.1.1 Theories Involving Stereo

1. A unified theory of generalized stereo vision (Larry Wolff [45, 49, 50]).
2. The derivation of shape from polarizing surfaces (Larry Wolff [46, 47, 48]).
3. Optimal estimators for stereo triangulation error (Ken Roberts, Dr. S. Kicha Ganapathy of AT&T Bell Laboratories [34]).

#### 0.1.2 Data Representations

1. A new representation for a line in three-space (Ken Roberts [35]).
2. Smooth interpolation of rotational motions (Ken Roberts, Drs. S. Kicha Ganapathy and Garry Bishop of AT&T Bell Laboratories [36]).

#### 0.1.3 Applications to Graphics

1. Realistic rendering of scenes using polarization properties (Larry Wolff, Dave Kurlander [51]).
2. A new data structure and algorithm for the mapping of arbitrary shapes (George Wolberg [43, 44]).

### 0.2 Middle-level Vision

#### 0.2.1 Regularized Surface Reconstruction and Stereo

1. A critical study of regularization methodology (Terry Boult [4, 10]).
2. Regularized surface reconstruction and segmentation based on smoothness energy (Terry Boult, Liang-Hua Chen [11]).
3. Integrated stereo matching, surface reconstruction, and surface segmentation (Terry Boult, Liang-Hua Chen [12, 16, 17]).

#### 0.2.2 Sensory Fusion

1. Fusion of multiple shape-from-texture methods (Mark Moerdler, John Kender [30, 31]).
2. Fusion of texture and stereo (Mark Moerdler, Terry Boult [5, 32, 33]).

#### 0.2.3 Shape from Dynamic Shadowing

1. A discrete method for deriving surfaces from dynamic shadows (John Kender, Earl Smith [26, 29]).
2. An optimal algorithm for shape from continuous shadows (Michalis Hatzitheodorou, John Kender [22, 23]).

#### 0.2.4 Application to Range Data

1. Recovery of superquadric parameters (Terry Boult, Ari Gross [6, 7, 13]).
2. Spline-based recovery of smooth oceanographic positional information (Terry Boult, Dr. Barry Allen of Columbia University's Lamont-Doherty Geological Observatory [8]).

### 0.3 Spatial Relations

#### 0.3.1 Representations of Objects and Space

1. Analysis and extension of issues in aspect graphs (John Kender, David Freudenstein on leave, Prof. Jonathan Gross [27]).
2. Survey of algorithms for the representation of space (Monnett Hanvey [21]).

[1]This work was supported in part by the Defense Advanced Research Projects Agency under contracts N00039-84-C-0165 and DACA76-86-C-0024.

3. Efficient updating of digital distance maps in dynamic environments(Terry Boult [9]).

#### 0.3.2 Theory and Practice of Navigation

1. Landmark definition and the representation and complexity of custom maps (John Kender, Avraham Leff [28]).
2. Systems issues in practical real-time robotic navigation (Monnett Hanvey, Drs. Bob Lyons and Russ Andersson of AT&T Bell Laboratories).

### 0.4 Parallel Algorithms

#### 0.4.1 Low- and Middle-level Vision Theory

1. Depth interpolation using optimal numerical analysis techniques on a pyramid machine (Dong Choi, John Kender [18, 19]).
2. Determination of surface orientation from foreshortened texture autocorrelations (Lisa Brown, Dr. Haim Shvaytser of Weizmann fellowship [14, 15]).

#### 0.4.2 Research and Applications on Tree Machines

1. Simulators and programming environments for Non-Von and for the Connection Machine (Hussein Ibrahim, Lisa Brown [20, 24, 25]).
2. Stereo, texture, and other pyramid-based algorithms (Hussein Ibrahim, Lisa Brown).

#### 0.4.3 Research and Applications on Pipelined Machines

1. Implementing basic real-time image algorithms for pipelined processors (Ajit Singh, Peter Allen [37, 38, 41]).
2. Sensor fusion of correlation and of spatio-temporal approaches to optic flow (Ajit Singh, Peter Allen, Dr. Surendra Ranganath of Philips Laboratories [39, 40, 42]).
3. Real-time object tracking and interception algorithms (Peter Allen [2]).

### 0.5 Robotics and Tactile Sensing

#### 0.5.1 System Development

1. Cartesian-based control of the newly-acquired Utah hand (Ken Roberts, Peter Allen).
2. Interfacing proprietary skin-like tactile sensors (Peter Allen).

#### 0.5.2 Multi-fingered Object Recognition

1. Sensor models and CAD/CAM object models (Peter Allen, Dino Tarabanis [1, 3]).
2. Haptic recognition via active exploration with a instrumented robot hand (Ken Roberts, Peter Allen)

We now detail these efforts, many of which are documented by full papers in these proceedings. We also include short discussions of work in progress.

## 1 Low-level Vision

We have extended our work on a generalized framework for the perception of physical surface properties, and have formalized the conditions under which image measurements can break certain symmetries of observation in order to uniquely define depth-related object values. We have applied this general theory to the specific case of the derivation of surface orientation from differences in the polarization of reflected light, and have shown that only two settings of a linear polarizer placed before the camera are necessary for uniqueness. In other work, we have analyzed the error in traditional two-camera parallax stereo using a Bayesian statistical approach, and have computed optimal estimators that are extensible to multi-camera imaging configurations.

Some of our work has lead to economies of data representation; in particular, we have discovered a new way of representing lines in three-space that requires only four parameters and is totally free of annoying special cases. In work on rotational motions, we have defined an efficient, closed form way of interpolating their representations as quaternions over the associated three-sphere; the method leads to surprisingly smooth animations.

Work on low-level vision often leads to corresponding results in graphics. We have empirically validated that our theory of polarization adds striking realism to the computer graphic generation of certain types of scenes involving reflections. Lastly, in the course of investigating efficient object tracking algorithms, we have devised and implemented a general but fast method for mapping arbitrary planar shapes onto each other, based on a new skeletonization data structure.

### 1.1 Theories Involving Stereo

#### 1.1.1 Generalized Stereo Vision

Generalized stereo begins as an abstract unification of two distinct exiting stereo techniques: traditional parallax stereo, which calculates surface depth by varying the camera focal point, and photometric stereo, which calculates surface orientation by varying the light positions. A generalized stereo method calculates arbitrary visual object features (world coordinate position, local surface orientation, Gaussian curvature, color reflectivity, etc.) by varying related physical imaging parameters (position of focal point, orientation of incident light source, polarization of incident light source, etc.) The object feature is determined by the intersection, in a parameter feature space, of solution loci generated from a system of equations relating features to parameters [45, 49, 50].

We have shown that in its formalized axiomatic definition, a generalized stereo method is characterized by four things: a visual object feature to be measured, a functional way of converting image observables such as image intensity into other observables such as image gradient, a set of variable imaging parameters, and the equations relating all three. We have illustrated the theory with many examples.

Additionally, we have characterized the error intrinsic to this family of methods by noting that the dimension of measurement ambiguity is readily determined by the implicit function theorem applied to the equations at the point of intersection. More accurately, error can be characterized in terms of symmetries of solution loci using the theory of groups. We have established two theorems which state the precise conditions under which the intersection of solution loci can be further disambiguated.

#### 1.1.2 Polarization Stereo

We have investigated several applications of the generalized stereo theory. We have analyzed how surface orientation can be calculated by varying the wavelength and/or linear polarization of a single incident light source [46]. More practically, we have also proposed a new technique to measure local surface orientation based on a more complete theory of reflection of light. This theory combines the Torrance-Sparrow theory of reflection with the Wolf polarization theory of "quasi-monochromatic" (monochromatically filtered) light [47, 48]. The technique enables surface orientation to be uniquely measured in arbitrary lighting by placing a simple monochrome filter and a linear polarizer in front of the sensor; two images taken at two orientations of the polarizer suffice. The equations that govern the calculations, called the polarization state matrix equations, are elaborate, but they are only a special case of the larger family of generalized stereo imaging equations.

#### 1.1.3 Optimal Stereo Triangulation Techniques

We have analyzed the positional error in stereo triangulation using a Bayesian statistical approach, and have derived optimal estimators based on several different sets of imaging assumptions [34]. One assumption models the camera error function in a new and more general way, by including a depth-sensitive (1/z^n) factor. Our techniques are elegantly extended to the case of more than two cameras.

Intuitively, we prove that the following methods are optimal. For a given stereo pair, reject any errors perpendicular to the epipolar line. Weight each camera's estimate of source point position by the reciprocal of the variance of its error function and the square of the depth of the source point from it. For more than two images, compute the results taking the images pairwise, then combine them by weighting each result by the square of the pair's baseline.

### 1.2 Data Representations

#### 1.2.1 Representation of Three-Space Lines

We have constructed a new representation for a line in Euclidean three-space which uses only four parameters, the minimal number allowable, and still avoids singularities and special cases [35]. Therefore, without sacrificing convenience of computation, it is no longer necessary to represent lines in the more traditional six-parameter forms (such as Plucker coordinates, or point-and-orientation form), although the new representation has the added advantage that it is easy to convert to those forms. The representation, involving two parameters for position and two for orientation, readily generalizes to Euclidean n-space, where it uses 2n-2 parameters.

#### 1.2.2 Interpolation of Rotational Motion

Smooth interpolation of rotational motion (as in a "perfect spiral" football pass) is important in computer animation, robot control, and hypothesis-guided computer vision. We have implemented a new, closed form algorithm for doing so, based on representing motions as quaternions on the unit three-sphere [36]. Resulting displays of interpolated values, and the computer animation sequences based on them, are smoother and more perceptually realistic than two existing methods.

### 1.3 Applications to Graphics

#### 1.3.1 Rendering Using Polarization Properties

We have applied the more complete theory of reflection developed in the work described above to ray tracing algorithms in computer graphics. We report striking differences in the rendering of certain scenes involving reflections when the phenomenon of polarization is included [51]; the differences are preferred by observers. This lends some evidence to the belief that the reflection model is at least qualitatively very correct.

#### 1.3.2 Mapping of Arbitrary Planar Shapes

In attempting to analyze and track objects between images, we discovered that the literature was silent on the problem of efficiently and smoothly mapping between two image regions which are delimited by arbitrary closed curves; such regions do not have the universally assumed four corners. We have specified and verified an algorithm that instead treats an image region as a collection of interior layers around a skeleton (similar to that in [43]). These layers impose a type of local polar coordinate system which allows each shape to be "unwrapped" into a tree-like representation. Region-to-region warping is then defined by a natural mapping between the two resulting trees [44]. Although there is no a priori way of defining quality of mapping, the results are esthetically pleasing.

## 2 Middle-level Vision

We have presented a critical overview of the regularization methodology, and have demonstrated new means of specifying the function class and its stabilizing functional that, although non-traditional, give qualitatively better results. We have exploited one of these ways, which is heavily dependent on the use of reproducing kernel-based splines, to surface segmentation; the method computes upper and lower bounds on local surface energy prior to surface labeling, and demonstrates good results on synthetic and real image and range data, and even on some transparent surfaces. Further, we have incorporated this energy-based approach into a system that integrates the formerly separate middle-level vision stages of stereo matching, surface reconstruction, and segmentation into a more straightforward one-step surface labeling based on a single measure of ambiguity; quantitatively, it results in a significantly higher percentage of correct matches.

We have designed, built, and verified on synthetic and real imagery, a blackboard-based system that fuses the independent and occasionally conflicting information from multiple (four or more) texture cues into a integrated method for surface segmentation and orientation determination; it is organized around a new image data structure, the augmented texel, and achieves sensor fusion via a Hough-like method on a trixelated Gaussian sphere. We have extended the method to a design and preliminary system (tested on a real image) that fuses the resulting surface orientation with the results of the one-step stereo method described above; this design thus coordinates the two intra-modality integrations with an inter-modality relaxation-based fusion of information through a weighted averaging, according to a non-traditional "smoothness norm", of zero-crossing and texel-centroid data.

Our work on the derivation of surface information from self-shadowing has resulted in a patent application for the discrete case. Additionally, we have analyzed the continuous case according to methods of functional analysis and have devised a provably optimal algorithm for surface recovery that is grounded in an unusual family of basis-like splines and an unusual iterative procedure for handling the non-linearity of the mutual illumination constraints; we have demonstrated a high degree of surface reconstructive accuracy on one-dimensional data.

Using a nonlinear least square minimization technique on the so-called inside-out function, we have designed and demonstrated a system for the robust recovery of superquadric parameters from both noisy synthetic and actual range data imagery, including even the case of a superellipsoid with negative volume (a construct used in solid modeling). Transferring our middle-level vision technology to a real world problem, we have begun to analyze various methods for inferring the geological structures below the surface of the ocean by first fusing several noisy sources of ship-board sensory data, such as satellite, dead reckoning, and gravitational information; this system for position tracking is now in regular field use.

## 2.1 Regularized Surface Reconstruction and Stereo

### 2.1.1 Critical Analysis of Regularization

We have presented a survey of some of the benefits promised by the regularization framework, and also of some of its difficulties, particularly the problems of determining appropriate functional classes, norms, and regularization stabilizing functionals [4]. When we subjectively tested (via established procedures of psychology) the results of the methodology applied to the surface reconstruction problem, we found that non-traditional formulations provided better results. It is not surprising that we were then able to document the lack of development of most of the promises of regularization theory, finding only three actual examples of its fruitful realization [10].

### 2.1.2 Energy-based Surface Segmentation

Although current surface reconstruction algorithms have strong foundations in mathematics, the segmentation aspects of the work are purely heuristic. We have developed and tested a non-heuristic algorithm which simultaneously reconstructs surfaces and segments the underlying data according to the same energy-based smoothness measure [11]. It is founded on the use of reproducing kernel-based splines, which allow efficient calculation of upper and lower bounds on the energy. The system naturally deals with occluded objects, and also with sharply slanted surfaces, such as roads as seen from a vehicle. We have verified the system on a gamut of artificial and natural data, including transparent surfaces.

### 2.1.3 One-step Stereo Matching, Reconstruction, and Segmentation

Traditional stereopsis is done in three phases: 1) suitable features are detected in each image, 2) corresponding features are matched and disparity is determined, and 3) a complete depth map is approximated and segmented. We have extended our work on non-heuristic segmentation by developing a new, one-step approach to stereopsis that unifies the stereo matching criteria with our already combined reconstruction and segmentation criteria [12, 16, 17]. The criteria is exploited in the form of a measure of match ambiguity, which is used to rank order all potential matches. The method results in fewer unmatchable features than the Marr-Poggio-Grimson method.

## 2.2 Sensory Fusion

### 2.2.1 Fusing Shape-from-Texture Methods

We continue to augment and refine our system for integrating various modalities for determining surface orientation from multiple, independent, conceptually parallel, and possibly conflicting textural cues. The system uses a new data structure, the augmented texel, which combines multiple constraints on orientation, each with its own assurity weighting, in a compact notation for a single surface patch [30, 31].

We have demonstrated the system using four texture modules (shape from spacing, eccentricity, orientation, and size), on both synthetic and real imagery of surfaces, some of them curved or transparent, with robust results: slant and tilt are usually recovered within a few degrees. We have shown examples of real surfaces for which individual texture methods fail to determine surface orientation accurately because of noise, but for which their fusion succeeds. Part of the noise tolerance of the system is derived from the relaxation refinement of initial hypotheses about surface orientation and extent, which themselves are derived from a (noise tolerant) Hough accumulation array on the surface of the trixelated Gaussian sphere.

### 2.2.2 Fusing Stereo with Texture

Having found ways of integrating into two separate processes the three steps of stereo perception and at least four methods of texture perception, we have combined our results in a single system that fuses stereo and texture together [5, 32, 33]. Although it is still under development (it has processed only a single real image), it is uniquely structured to provide two qualitatively different means of information fusion, namely, intra-process and inter-process integration. The latter incorporates a priori assumptions about surfaces, such as degrees and measures of smoothness, and communicates such data via a blackboard organization. Such a two-stage organization does not appear inconsistent with what is known about human visual modularization.

In particular, the stereo process uses the relative accuracy and sparseness of the centroid of texels to begin feature localization, later switching to traditional zero-crossings. The work is further characterized by the choice of smoothness measure; roughly it minimizes variation in the 1.5 derivative, not the second. Final integration is done by weighting the significance of a surface constraint produced by either process inversely proportionally to the total number of constraints the process outputs (otherwise stereo would always outweigh texture processing).

## 2.3 Shape from Dynamic Shadowing

### 2.3.1 The Discrete Case: Shape from Darkness

We have analyzed and validated on synthetic data a new method, called shape from darkness, for extracting surface shape information based on object self-shadowing under moving light sources [26]. Unlike most shape-from

methods, it does not require a reflectance map, and it works on non-smooth surfaces. Shadow information is stored in a novel data structure called the suntrace, which records the quantized angle of illumination at which a given image point was first illuminated. Given n points, the surface reconstruction problem becomes the satisfaction of 8n constraint equations in 2n unknowns, one unknown each for the upper and lower surface bound for each image point. An unusual form of relaxation, in which pixels can affect other pixels at a great distance, quickly converges to the solution. Columbia University has applied for a patent on the method [29].

#### 2.3.2 The Continuous Case: Optimal Shape from Shadows

We have analyzed the same problem in the continuous setting, decomposing the two-dimensional problem into a series of one-dimensional slices in the plane of the moving light source. Casting the problem in a Hilbert space, we derived a provably optimal algorithm which involves interpolating splines of an unusual piecewise linear form [22, 23]. A side system of inequalities is optionally invoked in order to preserve the implicit information that points interior to a shadowed region must lie below that shadow line. The problem has a natural parallelization, not only into slices, but also into hill-and-valley segments. Our implementation has demonstrated high accuracy using few light sources on even badly nondifferentiable test functions. We are now attempting to analytically determine optimal light source placement.

### 2.4 Application to Range Data

#### 2.4.1 Recovery of Superquadrics

Many have noted the simultaneous descriptiveness and compactness that superquadrics offer as a volumetric model; noted, too, is their well-defined inside-out functions needed in parameter recovery. However, we have determined that the primary concern in superquadric parameter estimation is the proper choice of the error-of-fit measures that control the nonlinear least square minimization techniques. We have explored the effectiveness of several such measures on many examples using noisy synthetic data and actual range images, including multiple views of the same object and a superellipsoid with negative volume, the latter being an important primitive for constructive solid geometry-based modeling. We have concluded that existing measures of fit are inadequate, and have proposed ones that perform better [6, 7, 13].

#### 2.4.2 Recovery of Oceanographic Positional Information

We have investigated the problem of integrating different types of positional information, such as various satellite and inertial data, in order to reconstruct the path taken by an exploratory geological ocean vessel. Typical paths are piecewise very smooth except at turns; the problem is therefore a one-dimensional analogue of the middle-level vision problem of smooth surface recovery from sparse depth data [8]. We further investigated the related two-dimensional analogue problem: the inference of the geological structures below the surface of the ocean floor from gravitational information. The problem was solved by again using smoothing splines, backed by a clever heuristic to ignore faulty outliers in the data. The system, with some amount of human intervention, has been put into regular use by the researchers at Columbia's Lamont-Doherty Geological Observatory.

## 3 Spatial Relations

We have extended the semantics of an object representation, called the aspect graph, to the more realistic imaging environments having finite camera precision; we fixed several definitional problems in the process. Having surveyed 60 papers on representations of navigational space, we have taxonomized the main approaches to the problem: they use dehydrated free space, simple mosaics, or reconstituted free space representations. We extended the path-planning representation, called the digital distance map, to dynamic environments, and presented an efficient algorithm for its maintenance under object movement.

We defined a model of landmarks and map-making, and showed that the problem of providing a navigator with a list of directions is, even in the simplest case, an NP-complete problem; nonetheless, heuristics exist to help cut down search in creating "good" maps (defined as being either "short" or "easy"). We have been programming a mobile robot platform, attempting to have it navigate topologically via landmarks such as corridor intersections.

### 3.1 Representations of Objects and Space

#### 3.1.1 Aspect Graphs and Degenerate Views

An aspect graph is a representation of the effect that viewing angle has on an object's observable features. While attempting to extend this concept to formations of objects, we discovered several inadequacies and errors in its current definition and use [27]. We demonstrated that the key concept of "characteristic view" is not well defined; in fact, it rarely is defined at all. The problem becomes more acute under finite camera resolution, where idealized aspect graphs become more like spatial maps: both nodes and arcs now have width. Given camera resolution and object size, we were able associate probabilities of observation to certain "degenerate views" of some simple objects.

Our most recent work has noticed a close connection between the aspect graph and the so-called first barycentric subdivision of standard graphs in graph theory; we are attempting to exploit this and other formal similarities.

#### 3.1.2 Representation of (Un)Occupied Space

We have completed a survey of some 60 papers dealing with environmental representations of mobile robots. Most of them assume a static two-dimensional world, and a complete bird's-eye knowledge of free space and obstacles. We have given a taxonomy of map primitives, such as frames of reference and map symbols, and a taxonomy of representations, such dehydrated free space (mixed polyhedra, and vertex graphs), simple mosaics (tessellations, distance maps, and quadtrees), and reconstituted free space (convex cells, and freeways). We have also noted the relative paucity of results on qualitative, topological navigation via landmarks.

#### 3.1.3 Dynamic Digital Distance Maps

A digital distance map contains in each of its cells information about the distance and/or direction to some pre-specified goal set; if the environment is static, it makes path planning trivial. We have developed an algorithm that extends the utility of these maps to dynamic environments, such as

robotic assembly domains [9]. The algorithm is efficient, in that it only updates those cells of free space that are in the moving object's "shadow", where a shadow is defined according to a precise but tricky grammar. The algorithm is two-phase: shadow calculation followed by map update; if the robot can avoid shadows, this allows some parallelization of robot movement with map updating. We have observed speedups of 25 times over brute force update. We are now extending the method to higher dimensions, such as configuration spaces, and investigating the use of multi-resolution techniques.

### 3.2 Theory and Practice of Navigation

#### 3.2.1 The Computational Complexity of Map-Making

We have formalized a model of topological navigation in one-dimensional spaces, such as along single roads, corridors, or transportation routes, and have shown that the problem is surprisingly difficult computationally [28]. In our model, we carefully discriminated between the concepts and representations of the world itself (a version of "Lineland"), the world as abstracted into symbols and landmarks by an omniscient map-maker, and the world as experienced by a limited navigator who follows the map-makers directions. Having also modeled the navigator's sensors in a primitive way (a sensor here being more like a feature detector), we proved that the problem of choosing an effective and efficient subset of sensors for navigation via landmarks is NP-complete.

However, the A* search procedure does apply; we also gave a simplifying heuristic evaluation function ("most new eliminated objects") for use with it. Having selected the proper sensor subsets, Dijkstra's shortest path algorithm gives the optimal set of directions for the navigator, where we defined an optimal map to be one that minimizes length of directions, cost of sensing, or some combination.

#### 3.2.2 Driving the AT&T Mobile Robot

In work jointly supported by AT&T Bell Laboratories, we are investigating several systems issues in navigation by using their mobile robot platform. As an early experiment in landmark recognition, we have programmed it to track walls with its sonar; the robot notices intersections and dead ends, which are potentially significant external environmental cues for self-positioning. In related work, we have tackled the problem of calculating ranges to visually perceived vertical edges by using a simplified Kalman filter. Since the error introduced by quantization and other factors is not gaussian, this filter produces accurate estimates only at selected points; however, these estimates can be strategically combined using triangulation to increase accuracy. We are testing this filtering/triangulation system on the robot, aiming for 60 Hz cycle time.

## 4 Parallel Algorithms

We have analyzed the performance of the parallelization of several computationally optimal algorithms for depth interpolation, and have found that on a wide variety of synthetic and real range data the adaptive Chebyshev is the most efficient, even when implemented in a multigrid fashion. We have invented a particularly simple, accurate, and robust shape-from-texture algorithm based on image autocorrelation that outperforms human observation on real scenes of roads, dirt, and grass.

In our Strategic Computing work, we have designed and implemented three related programming environments for validating parallel pyramid-based SIMD algorithms; the third one elegantly exploits the Connection Machine's hypercube network to efficiently emulate a library of image functions for a virtual pyramid machine, at a fixed low overhead. We have also implemented, on simulators or the emulator, parallel pyramid-based stereo, segmentation, and Hough algorithms, as well as our new autocorrelation texture algorithm.

On our PIPE, we have built up a basic library of pipelined low-level image processing functions. We are implementing a system for optic flow determination that fuses the results of intensity correlation methods and spatiotemporal energy methods; the latter is based on a novel image structure called the pyramid of oriented edges. The PIPE is fast enough to provide real-time robot arm control information, which we are preparing to demonstrate by the dynamic grasping of moving objects.

### 4.1 Low- and Middle-level Vision Theory

#### 4.1.1 Optimal Depth Interpolation

Many constraint propagation problems in early vision, including depth interpolation, can be cast as solving a large system of linear equations where the resulting matrix is symmetric, positive definite, and sparse. We have analyzed and simulated several numerical analytic solutions to these equations for a fine grained SIMD machine with local and global communication networks (e.g., the Connection Machine); the methods are provably optimal in terms of computational complexity. We have established that for a variety of synthetic and real range data, the adaptive Chebyshev acceleration method executes faster than the conjugate gradient method, if near-optimal values for the minimum and maximum eigenvalues of the iteration matrix are available [18].

We then extended these iterative methods by implementing them in a pyrmidal multigrid (coarse-medium-fine) fashion. Again we showed that, when used with a fixed multilevel coordination strategy, the multigrid adaptive Chebyshev acceleration method executed faster than the multigrid conjugate gradient method [19]. Further, we demonstrated that because an optimal Chebyshev acceleration method requires local computations only, it in turn executes faster than either adaptive Chebyshev acceleration or conjugate gradient methods, both of which require global computations. We continue to validate these algorithms on Utah laser range data.

#### 4.1.2 Shape from Texture Autocorrelation

We have developed a new method for determining local surface orientation from rotationally invariant textures based on the two-dimensional two-point autocorrelation of an image [14, 15]. This method is computationally simple and easily parallelizable, uses information from all parts of the image, assumes only texture isotropy, and requires neither texels nor edges in the texture; it is thus more widely applicable than the method of Witkin. We have demonstrated that when applied to locally planar patches of real textures such as roads, dirt, and grass, the results are highly accurate, even in cases where human perception is so difficult that people must be assisted by the presence of an artificially embedded circular object.

Along the way, we have proven that the algorithm has several exploitable mathematical elegancies. For example,

the autocorrelation of an isotropic texture will always be entirely composed of concentric scaled elliptic iso-contours; this makes the extraction of slant and tilt values from ellipse parameters nearly trivial. Secondly, because the autocorrelation has such robust structure, it is easy to filter out from it spurious noise such as that commonly generated by the horizontal smearing of pixels in typical CCD cameras.

## 4.2 Research and Applications on Tree Machines

### 4.2.1 Simulators and Programming Environments

As part of our efforts under Strategic Computing, we have developed three programming environments that support our research on stereo and texture algorithms, in parallel image pyramid style [20]. Our first environment assisted work on the fine-grained tree-structured SIMD Non-Von supercomputer (now discontinued) [24], and it consisted of simulators of various grain sizes. As Non-Von began winding down, we constructed a second, more abstract environment of image function primitives for general pyramid machine vision work; this environment was necessarily a transitional one for the preservation of the prior work. For our third and current programming environment, we have designed, installed, and documented a highly efficient pyramid machine emulator that executes those image function primitives on the University of Syracuse Connection Machine [25].

This third environment cleverly reduces communication contention by an elegant embedding of the pyramid within the hypercube network. Mesh operations take only a small fixed amount of overhead proportional to the size of the hypercube; parent/child operations run in a smaller fixed time independent of hypercube size. The image functions allow the user to create pyramid data structures, to load/unload various pyramid levels, to move data up/down, and to perform several operations such as convolution and hierarchical operators on the created data structures. Both our texture and stereo work will benefit from the multiresolution capabilities: texture algorithms will adjust to texel size, and stereo will use feature-matching based on hierarchical correlation. Most recently, we are upgrading our environment to run on the CM2.

### 4.2.2 Pyramid-based Stereo and Texture

Our main objective under Strategic Computing is to develop, implement, and integrate parallel multi-resolution stereo and texture algorithms for determining local surface orientation and depth, to be used by autonomous land vehicle navigation systems.

We have implemented on the Non-Von simulator a straightforward parallelization of a multi-resolution version of the Marr-Poggio stereo algorithm, which achieves some economies on the SIMD architecture by exploiting the eight-fold symmetries of digitized Laplacian of Gaussian masks. We are parallelizing our new autocorrelation-based shape-from-texture technique for the Connection Machine, where it becomes technically even more elegant. Using image shifts to compute a finite window of the autocorrelation, we can compute surface orientation for surface patches in constant time.

In service to both of the above algorithms, we have implemented two parallel texture-based image segmentation algorithms and tested them on ERIM ALV road data. The first algorithm uses micro-edge density counts in dynamically varying windows that attempt to track the road edge from image to image; success has been limited by the low texture resolution. A second algorithm is both parallel and pyramid based, and is more successful. It constructs up-down links in the multi-resolution pyramid between nodes on adjacent levels, according to similarities of spatial grey level dependence statistics. By top-down iterative refinement of these links, the pyramid and hence the image is segmented; basically this is a parallel form of region-splitting. Lastly, we have implemented the generalized Hough transform, including the parallel creation of the reference tables.

## 4.3 Research and Applications on Pipelined Machines

### 4.3.1 Real-time Algorithm Library

We are developing real-time "pixel-parallel" versions of a variety of image processing algorithms for our PIPE architecture. Based on our past experience with pipelined processors [41], we have already installed algorithms for spatial filtering, spatiotemporal filtering, and pyramid-based spatial processing [37]. Representative examples include edge preserving smoothing, generalized n-by-n convolution, normal optic-flow, thinning and morpological operators, and the pyramid representations of Burt, Mallat, and Singh and Ranganath [38].

### 4.3.2 Motion Perception Sensor Fusion

In work that is jointly supported by Philips Laboratories [39], we are implementing a multi-sensor fusion approach to the robust measurement of optic flow. Via confidence measures, we are integrating intensity correlation methods, which work best in structured scenes, and spatiotemporal energy methods, which are more suited for textured scenes [40, 42].

The spatio-temporal frequency approach is implemented on the PIPE using a pyramid image structure, called Pyramid of Oriented Edges; the POE is a logical extension of Burt's and of Mallat's pyramids, both of which we have also implemented. Using the POE, we have extracted coarse optic flow fields for a number of real images. We plan to extend the method by developing a hierarchical set of spatio-temporal frequency-tuned filters which will extract true optic flow from the POE data, and then integrate the results with our implementation of an intensity correlation-based model similar to that of Anandan.

### 4.3.3 Real-time Motion Tracking

The 60 Hz frame-rate image processing abilities of our PIPE enable it to generate visual tracking information fast enough to be coordinated with the motion control of a robotic arm. We have implemented pipelined algorithms to perform motion detection, thinning, and region-of-interest segmentation in order to track objects with a wrist-mounted camera in real-time [2]. Most of the processing is pyramid-based, and uses spatio-temporal filters. We have also implemented a motion-control process that concurrently calculates on the Masscomp host the predicted trajectory for the moving part, in order for the arm to intercept it for grasping.

# 5 Robotics and Tactile Sensing

We have recently acquired a Utah/MIT dextrous hand, for which we are developing tactile control algorithms. Having also recently acquired proprietary sensing "skin", we are also building its interface electronics and software. Through low-level sensor models and CAD/CAM object models, we continue to pursue the automatic generation of sensing

constraints and strategies. We have begun to investigate the features, representations, and primitive operations necessary for the recognition of objects under multi-fingered and multi-jointed active haptic exploration.

### 5.1 System Development

#### 5.1.1 Low-level Control

The Utah/MIT dextrous hand has provided us with a new set of tools to continue our study of intelligent touch and grasping. We are implementing Cartesian-based low level control algorithms for the hand, and extending them to a more hybrid scheme using both tendon force and tactile contacts.

#### 5.1.2 Conformable Tactile Pads

We are currently implementing tactile sensors for each of the hand's fingers, using a proprietary piezoelectric polymeric material similar to that used on electronic drum pads. This responsive, conformable skin-like material can be deposited on arbitrary surfaces, and has extraordinarily good signal isolation and hysteresis characteristics. We are building a multiplexor for the sensor signals, in the hope of achieving real-time integration of the tactile sensor feedback into the low level hand control loop.

### 5.2 Multi-fingered Object Recognition

#### 5.2.1 CAD/CAM and Sensor Models

Our experience with merging multiple sensor data sources [1, 3] has led us to examine the sensor modeling problem from the perspective of the automatic generation of viewpoint, geometric, and sensing constraints. We assume an assembly or an inspection domain, and our analysis is based on both CAD/CAM object models and low-level sensor models. The emphasis is on the automatic and intelligent handling of partial object descriptions and partial or total sensor occlusions. The sensors we model are, among others, monocular and binocular camera systems, laser range finders, and several types of active touch sensors.

#### 5.2.2 Active Haptic Exploration

We have begun to analyze active multi-finger touch strategies to recognize objects haptically: that is, by only using external tactile sensors and internal force and position sensors. We are investigating the necessary data structures, procedural organizations, object models, and feature constraints that are the necessary prerequisites to active exploration; they may overlap with tools and methods in vision. We propose to demonstrate our haptic understanding of an object by establishing a secure enough grip to lift it.

## 6 References


**1.** Allen, P.K.. *Robotic Object Recognition Using Vision and Touch.* Kluwer Academic Publishers, 1987.

**2.** Allen, P.K. Real-Time Motion Detection on a Frame Rate Processor. Extended Abstracts of the 41st Annual SPSE Conference, May, 1988. (To appear).

**3.** Allen, P.K. "Integrating Vision and Touch for Object Recognition Tasks". *International Journal of Robotics Research* (To appear 1988).

**4.** Boult, T.E. What is Regular in Regularization? Proceedings of the IEEE First International Conference on Computer Vision, June, 1987, pp. 457-462.

**5.** Boult, T.E., and Moerdler, M.L. An Experimental System for the Integration of Information from Stereo and Multiple Shape-from-Texture Algorithms. Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision VI, November, 1987.

**6.** Boult, T.E., and Gross, A.D. Recovery of Superquadrics from Depth Information. Proceedings of the AAAI Workshop on Spatial Reasoning and Multi-Sensor Fusion, October, 1987, pp. 128-137.

**7.** Boult, T.E., and Gross, A.D. Recovery of Superquadrics from Depth Information. Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision VI, November, 1987.

**8.** Boult, T.E. "Optimal Algorithms: Tools for Mathematical Modeling". *Journal of Complexity 3* (1987), 183-200.

**9.** Boult, T.E. Updating Distance Maps When Objects Move. Proceedings of the SPIE Conference on Mobile Robots II, November, 1987.

**10.** Boult, T.E. Regularization: Problems and Promises. Extended Abstracts of the 41st Annual SPSE Conference, May, 1988. (To appear).

**11.** Boult, T.E., and Chen, L.-H. Energy-Based Surface Segmentation. Department of Computer Science, Columbia University, February, 1988.

**12.** Boult, T.E., and Chen, L.-H. Analysis of Two New Stereo Algorithms Integrating Surface Reconstruction and Matching. Department of Computer Science, Columbia University, February, 1988.

**13.** Boult, T.E., and Gross, A.D. On the Recovery of Superellipsoids. Proceedings of the DARPA Image Understanding Workshop, April, 1988. (These proceedings).

**14.** Brown, L.G., and Shvaytser, H. Surface Orientation from Projective Foreshortening of Isotropic Texture Autocorrelation. Proceedings of the DARPA Image Understanding Workshop, April, 1988. (These proceedings).

**15.** Brown, L.G., and Shvaytser, H. Surface Orientation from Projective Foreshortening of Isotropic Texture Autocorrelation. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June, 1988. (To appear).

**16.** Chen, L.-H., and Boult, T.E. An Integrated Approach to Stereo Matching, Surface Reconstruction, and Depth Segmentation Using Consistent Smoothness Assumptions. Proceedings of the DARPA Image Understanding Workshop,

April, 1988. (These proceedings).

**17.** Chen, L.-H., and Boult, T.E. Analysis of Two New Stereo Matching Algorithms. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June, 1988. (To appear).

**18.** Choi, D.J., and Kender, J.R. Solving the Depth Interpolation Problem on a Parallel Architecture. Proceedings of the IEEE Computer Society Workshop on Computer Architectures for Pattern Analysis and Machine Intelligence, October, 1987, pp. 107-114.

**19.** Choi, D.J., and Kender, J.R. Solving the Depth Interpolation Problem on a Parallel Architecture with a Multigrid Approach. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June, 1988. (To appear).

**20.** Gross, A.D., and Rosenfeld, A. "Multiresolution Object Detection and Delineation". *Computer Vision, Graphics, and Image Processing 39* (1987), 102-115.

**21.** Hanvey, M. Environmental Representations for Mobile Robot Navigation. Department of Computer Science, Columbia University, January, 1988.

**22.** Hatzitheodorou, M., and Kender, J.R. An Optimal Algorithm for the Derivation of Shape from Shadows. Proceedings of the DARPA Image Understanding Workshop, April, 1988. (These proceedings).

**23.** Hatzitheodorou, M., and Kender, J.R. An Optimal Algorithm for the Derivation of Shape from Shadows. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June, 1988. (To appear).

**24.** Ibrahim, H.A.H, Kender, J.R., and Shaw, D.E. "Low-Level Image Analysis Tasks on Fine-Grained Tree-Structured SIMD Machines". *Journal of Parallel and Distributed Computing 4* (1987), 546-574.

**25.** Ibrahim, H.A.H. Pyramid Algorithms Implementation on the Connection Machine. Proceedings of the DARPA Image Understanding Workshop, April, 1988. (These proceedings).

**26.** Kender, J.R., and Smith, E.M. Shape from Darkness: Deriving Surface Information from Dynamic Shadows. Proceedings of the IEEE First International Conference on Computer Vision, June, 1987, pp. 539-546.

**27.** Kender, J.R., and Freudenstein, D.G. What is a Degenerate View? Proceedings of the Tenth International Joint Conference on Artificial Intelligence, August, 1987, pp. 801-804.

**28.** Kender, J.R., and Leff, A. On the Computational Complexity of Linear Navigation. Proceedings of the DARPA Image Understanding Workshop, April, 1988. (These proceedings).

**29.** Kender, J.R., and Smith, E.M. A Method and an Apparatus for Determining Surface Shape Utilizing Object Self-Shadowing. Patent Application 058,914. July, 1987.

**30.** Moerdler, M.L., and Kender, J.R. An Integrated System that Unifies Multiple Shape-from-Texture Algorithms. Proceedings of the Sixth National Conference on Artificial Intelligence, July, 1987, pp. 723-727.

**31.** Moerdler, M.L., and Kender, J.R. An Approach to the Fusion of Multiple Shape-from-Texture Algorithms. Proceedings of the AAAI Workshop on Spatial Reasoning and Multi-Sensor Fusion, October, 1987, pp. 272-281.

**32.** Moerdler, M.L., and Boult, T.E. The Integration of Information from Stereo and Multiple Shape-from-Texture Cues. Proceedings of the DARPA Image Understanding Workshop, April, 1988. (These proceedings).

**33.** Moerdler, M.L., and Boult, T.E. The Integration of Information from Stereo and Multiple Shape-from-Texture. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June, 1988. (To appear).

**34.** Roberts, K.S., and Ganapathy, S.K. Stereo Triangulation Techniques. Proceedings of the IEEE Computer Society Workshop on Computer Vision, November, 1987, pp. 336-338.

**35.** Roberts, K.S. A New Representation for a Line. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June, 1988. (To appear).

**36.** Roberts, K.S., Ganapathy, S.K., and Bishop, G. Smooth Interpolation of Rotational Motions. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June, 1988. (To appear).

**37.** Singh, A. Image Processing on PIPE. Robotics and Flexible Automation Department, Philips Laboratories, June, 1987.

**38.** Singh, A., and Allen, P.K. Constructing Multi-Resolution Image Representations in Real Time. Department of Computer Science, Columbia University, July, 1987.

**39.** Singh, A., and Ranganath, S. A Hierarchical Model for Directional Selectivity in Space and Measurement of Optic-Flow. Robotics and Flexible Automation Department, Philips Laboratories, August, 1987.

**40.** Singh, A. Measurement of Visual Motion. Department of Computer Science, Columbia University, October, 1987.

**41.** Singh, A., and Lala, P.K. "A Multilayer Cellular Architecture for a Highly Parallel VLSI Supercomputer". *IEEE Transactions on Computers* (To appear 1988).

**42.** Singh, A., and Allen, P.K. A Real-Time Hierarchical Model for Optic Flow Determination via Spatiotemporal Frequency Channels. Proceedings of the DARPA Image Understanding Workshop, April, 1988. (These proceedings).

**43.** Wolberg, G. "A Syntactic Omni-font Character Recognition System". *International Journal of Pattern Recognition and Artificial Intelligence 1* (1987), 303-322.

**44.** Wolberg, G. Image Warping Among Arbitrary Planar Shapes. Proceedings of the Computer Graphics International Conference, May, 1988. (To appear).

**45.** Wolff, L.B. A General Formalization of Stereo Vision. Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision VI, November, 1987.

**46.** Wolff, L.B. Spectral and Polarization Stereo Methods Using a Single Light Source. Proceedings of the IEEE First International Conference on Computer Vision, June, 1987, pp.

708-715.

**47.** Wolff, L.B. Surface Orientation from Polarization Images. Proceedings of the SPIE Conference on Optics, Illumination, and Image Sensing for Machine Vision II, November, 1987.

**48.** Wolff, L.B. Shape from Polarization Images. Proceedings of the IEEE Computer Society Workshop on Computer Vision, November, 1987, pp. 79-87.

**49.** Wolff, L.B. Towards a More General Conception of Stereo Vision. Proceedings of the AAAI Spring Symposium Series, March, 1988.

**50.** Wolff, L.B. An Introduction to Generalized Stereo Techniques. Proceedings of the DARPA Image Understanding Workshop, April, 1988. (These proceedings).

**51.** Wolff, L.B. Ray Tracing Using Polarization Parameters. Department of Computer Science, Columbia University, January, 1988.

# KNOWLEDGE-BASED VISION TECHNOLOGY

# PROGRESS AT HUGHES AI CENTER

K.E. Olin, M.J. Daily, J.G. Harris, K. Reiser

Hughes Research Laboratories, AI Center
23901 Calabasas Road, Calabasas, CA 91302

***Introduction.*** The research progress at Hughes AI Center reported in these proceedings is primarily concerned with two topic areas: first, vision and systems for navigation, and second, object recognition.

The first cross-country map and sensor-based autonomous operation of a robotic vehicle in natural terrain was achieved in August 1987 and repeated with extended capabilities in December 1987.† The vehicle avoided difficult obstacles such as bushes, gullies, rock outcrops, and steep slopes as seen in Figure 1. This success was attained by Hughes through a series of experiments performed on the Autonomous Land Vehicle (ALV) at the Martin Marietta Aerospace Corporation (MMAC), Denver test site. In preparation for experiments on-board the ALV an extensive simulation capability was developed. In Section 1, we present an overview of our approach, the perception system development, and the planning system. Details of the perceptı ı and planning techniques are presented in accompanying papers [1,2].

A multiresolution image interpretation system was developed for object detection. The objective of this system was to detect objects by reasoning with multiple feature sets represented in object models. A feature set is comprised of a rich variety of image primitives together with spatial relationships. The multiresolution interpretation used lower resolutions to focus attention and higher resolutions for object details. An overview of this system is presented in Section 2; a more thorough discussion is found in an accompanying paper by T. M. Silberberg [3].

**Figure 1.**
**Typical cross-country terrain for the ALV**

## 1.0 Autonomous Land Navigation

### 1.1 Problem

Compared to structured environments, navigation in an unstructured environment such as cross-country terrain presents a significantly different set of problems for a perceptual system. While recognition of man-made objects performs adequately by utilizing rectilinear surface features, color characteristics, or other well constrained surface properties, there are no comparable structural expectations available in natural terrain. It is reasonable for indoor mobile robots to project a route using a mobility map that includes obstacles identified as occupying vertical space and assumes that the floor is flat. However, meaningful objects in natural terrain are more diverse and difficult to characterize. Specific features such as local slope, three dimensional edges, or color of specific regions may or may not represent obstacles. In general, the features are difficult to extract and difficult to combine to form reliable descriptions of the terrain. For example, trees can have widely varying shapes, colors, and sizes making tree recognition in itself a formidable problem The recognition problem is compounded by seasonal and weathering effects, and the need to interpret the environment in a timely manner.

### 1.2 System Architecture

The goal of the Knowledge Based Vision Techniques (KBVT) research at the Hughes AI Center is to provide the necessary perceptual descriptions of the environment which allow an autonomous vehicle to successfully navigate amongst obstacles. It is also our goal to transfer technology and perform experiments on-board a vehicle. The latter goal has constrained our research in the past year to approaches and algorithms which could "perceive" obstacles in a timely manner; that is, the perception processing has to allow ample time for the vehicle to safely respond to obstacles. An important feature of the perception system is its evolution in conjunction with the planning system also developed by Hughes AI Center.

† Development of this system has been supported by Defense Advanced Research Projects Agency (DARPA) contract #DACA76-85-0007. Progress on a coordinated DARPA contract #DACA76-85-0017 is also reported.

Our approach defines a system architecture which supports a decomposition of the problem directed by immediacy and assimilation considerations [4,5]. A brief overview of the architecture is included as an introduction to the technology development and experimentation results.

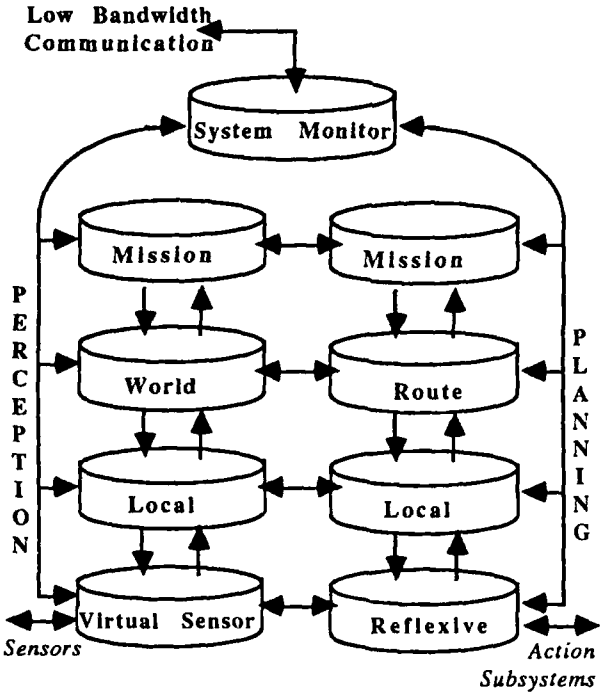



**Figure 2. System Architecture**

In this hierarchical control structure as shown in Figure 2, the vertical structure generally reflects the level of information fusion. Higher levels function on the basis of highly assimilated data that are generally symbolic and require longer interpretation time and larger spatial area; the lower levels exploit more immediate data. A failure at one of the lower levels is signalled to the next higher level, which then re-assesses the situation and adjusts accordingly.

For the ALV experiments a subset of the hierarchical control system was used. A simple mission was defined by start and goal locations with the path constrained to maintain a direct line-of-sight with the communications tower. At the *route* level of the hierarchy a map-based planner was used to generate all experimental routes. A route consisted of a set of subgoal points. Our major emphasis in the past year was to develop and experiment with the lowest level of the hierarchy. At this level, *virtual sensors* and *reflexive behaviors* (or behaviors) are used as the real-time operating primitives for the rest of the system. Knowledge assimilation is minimized in order to provide the fastest possible vehicle response. Virtual sensors are sensing and processing units that detect specific environmental features and relay information about features to the reflexive behaviors. A virtual sensor is "contracted" by a behavior to provide information at a required processing rate and accuracy. The reflexive behaviors are highly procedural units that operate on virtual sensor data to provide real-time control. Virtual sensors and reflexive behaviors are grouped into *activities* so that multiple behaviors can operate concurrently to produce control decisions. These activities are scheduled by the local planner to achieve current goals.

No single reflexive behavior and virtual sensor combination is ever expected to be able to handle vehicle navigation problems in general; but rather, several are used in conjunction, each designed to handle a specific sub-problem within the overall range of navigation tasks. It is the responsibility of the local perception and planning modules to guarantee that the selected activities are appropriate for the current environment. Our recent experimentation on-board the ALV has shown the concept of virtual sensors and reflexive behaviors provide a viable approach to local vehicle control.

### 1.3 Obstacle Detection

In the context of cross-country navigation, we have defined an obstacle as any region a particular vehicle cannot traverse. This definition allows obstacle detection to depend upon the mobility characteristics of the vehicle. Therefore, a natural approach to perception for cross-country navigation models the vehicle's interaction with a three dimensional representation of the sensed terrain to determine traversability. The ALV is equipped with a laser range scanner which measures the distance along the line of sight to the nearest object. This sensor inherently supplies information of surface geometries; however, the interpretation of this information is difficult. Several methods to interpret surface geometry in terms of an ALV mobility model were developed that employ a down-looking Cartesian Elevation Map (CEM). In addition, methods were developed to weight obstacles such that potential paths are penalized in areas "rich" with obstacles. These methods are briefly described in the following discussion. We also mention briefly parallel considerations and implementations. A separate paper is recommended for a more complete description of these techniques [1].

Cartesian Elevation Map

The Cartesian Elevation Map (CEM) is a representation for range information in which data from the viewer centered coordinate system of the sensor is transformed into a Cartesian $z(x,y)$ coordinate system. This results in a down-looking map view representation of terrain which is useful for autonomous navigation. This same representation may be obtained from other depth sensors, such as stereo or sonar.

The development of the CEM required us to deal with a variety of range processing issues. As one would expect, the elevation data represented in the CEM are highly oversampled in the immediate foreground and undersampled at greater distance from the sensor. In addition, there are some regions in the CEM that fall outside the field of view of the scanner and other regions that fall behind (in the shadows of) tall features in the terrain. In the CEM, areas with sufficiently dense sampling of points are fitted with a continuous surface and undersampled areas are explicitly labeled "unknown".

Due to the limited vertical field of the laser range scanner (30 degrees), terrain immediately in front of the sensor is not visible. The closest scanned ground in our experiments is approximated 13 feet in front of the vehicle. We have investigated the fusion of data from previous CEM's to fill in this unknown area. One method uses the orientation sensors on board the ALV (heading, pitch, roll, x, and y) together with an estimate of change in the z-position to determine vehicle motion in all six degrees of freedom. We are currently investigating another method that recovers the motion directly from sequences of range images.

Vehicle Model Trajectory Method

We have developed a relatively sophisticated three dimensional model of the ALV. This model together with the CEM yields a formal definition of obstacles, thus avoiding *ad hoc* and incomplete definitions. The model is currently represented by minimizing the energy of the suspensions springs associated with each wheel as the vehicle is applied at a position and orientation in the CEM. Three types of obstacles are detected with the vehicle model: suspension, slope, and clearance as shown in Figure 3. This definition of obstacles has performed fairly reliably in our experiments on the ALV. The model will be extended to include constraints such as vehicle weight distribution, weather conditions, risk factors, and vehicle speed and dynamics.

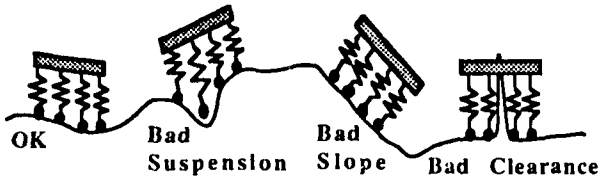



**Figure 3.**
**Obstacle Definitions Using the Vehicle Model**

The vehicle model allows us to produce a three dimensional traversability map by applying the model at each possible position and heading. In most cases, a complete traversability map of the entire sensed area is not needed. Because of constraints on perception processing time, we have developed techniques for applying the model only at those points necessary to fulfill requests issued from the planner. This method, called the Vehicle Model Trajectory (VMT) virtual sensor, simply calculates the projected heading of the vehicle at each point along a linear trajectory and applied the model at that heading and location until it either reaches the end of the trajectory or assumes an unstable configuration. The virtual sensor is contracted with the behaviors to return the distance travelled (or "safe distance") and the reason for termination. The VMT virtual sensor can provide different levels of accuracy and processing speed by varying three parameters independently: CEM size and resolution, vehicle model accuracy, and sampling frequency of the range image.

In the recent ALV experiments, the VMT virtual sensor was used along seven linear trajectories. The total processing time from image acquisition to trajectory output was approximately six seconds on a Sun 3 with a floating point coprocessor.

**Figure 4.**
**Cartesian Elevation Map and Vehicle Model Trajectories**

The current implementation of the VMTs uses only linear trajectories although the Hughes planner controls the vehicle through speed and turn rate commands which result in curved trajectories. We plan to generalize the VMT virtual sensor for any given trajectory; Figure 4 shows curved trajectories. Other enhancements are discussed in [1].

Cartesian Weight Map

The current VMT strategy simply tells the planner how far the vehicle may go along a given trajectory. The planner has no idea how close the trajectory passes by an obstacle. In simulation and in real life, the vehicle tends to pass too close to obstacles since no part of the current system deals with side clearance.

Like the CEM, the Cartesian Weight Map (CWM) is a down-looking map in the local coordinate system of the vehicle. A pixel value in the CWM represents the weight identified with the cost of traversing the corresponding pixel in the CEM. Nontraversable obstacles found using a Gradient of Gaussian (GoG) technique and verified by the vehicle model are given an infinite weight in the CWM so that no path will ever travel through an obstacle. To solve the problem of the vehicle passing too close to obstacles, all other CWM pixels are given weights which exponentially decay with distance from the nearest obstacle.

The CWM could be extended to penalize areas which are "bumpy" and reward regions that are smooth. This concept of multiple virtual sensors concurrently updating the CWM is consistent with our hierarchical system design. We have tested the CWM in simulation and found that the technique safely guides the vehicle equidistant between obstacles and avoids small cul-de-sacs.

Parallelism

A great advantage of the CEM and CWM methods is that they are extremely parallel. In cooperation with MIT in March, 1987, we implemented (during a two week

programming spree) the CEM construction algorithms and the GoG detection technique on the Connection Machine. Creating a 128x128 CEM from a range image took less than 0.5 seconds with unoptimized Lisp code and intensive floating point calculations; the GoG required an additional 8 milliseconds of compute time. We expect these times will be significantly reduced with the CM-2.

In addition, we have explored implementing the CEM on a WARP. Since the pixel in the range image that contributes to a given location in the CEM is data-dependent, the entire range image is distributed to each of the 10 cells in the WARP array. The CEM is then divided into 10 column swaths with several columns overlap between each swath. Each WARP cell processes every pixel in the scan, but saves only those points that fall within its assigned column swath. Progress on the WARP implementation has been delayed while we await the new release of WPE 2.6.

An implementation design similar to the WARP implementation has been designed for the Hughes Hierarchical Bus Architecture (HBA) [6]. The HBA is available for use in the lab simulation environment to improve simulation throughput.

### 1.4 Obstacle Avoidance

The planning system for obstacle avoidance is designed to provide real-time vehicle control while maintaining the flexibility needed for operation in realistic environments. Because the primary objective of the cross-country navigation experiments was to test critical real-time perception and planning interfaces, the map-based and reflexive behavior modules were the primary focus for development. More detail of the planning system are included in these proceedings [2].

Map-based Planning

At the route level of the hierarchy resides the map-based planner providing route information obtained from digital terrain maps. The route planning data includes maps of landcover, elevation, hydrology, roads, and landforms, and also data, such as visibility from the vehicle to the communications shell, that was derived from these maps. The map-based planner generates all experimental routes; a route consists of a set of subgoal point locations.

Reflexive Planning

The reflexive planning module is tasked with achieving the subgoals received from the map-based planner. The execution of the path requires the planner to react to perceived information, but do so in a consistently reasonable manner (i.e., intelligently). Reflexive planning controls the vehicle within its immediate environment with minimal data assimilation.

The behaviors used in the ALV experiments respond to VMT virtual sensor. The interface between the perception, planning, and vehicle control is critical. For instance, planning to avoid an "unknown" area changes dynamically as more information is perceived or as known obstacles are detected. In addition, as obstacles pass below the range scanner field of view a method is necessary to determine when the vehicle has traveled completely beyond the obstacle bounds. Other timing related issues must also be addressed; such as, when the vehicle reaches the end of a VMT it must slow down and stop if necessary to wait for new VMT data.

For the first set of experiments, the behaviors were grouped into two activities: the first activity was designed to travel toward a goal when the vehicle was in a clear area, and the second activity was designed to control the vehicle when obstacles were present. For the recent experiments, the behaviors were incorporated into a single activity that used a technique which weighted the importance of the goal according to the difficulty of the terrain. This is desirable because as the vehicle's movement becomes more restricted it becomes more important to get clear of the rough area than to make progress directly toward the goal. In the case when the vehicle is in an area free of obstacles, the goal weight becomes predominate and the vehicle tends to head straight toward the goal.

### 1.5 Simulation Environment

Simulation plays a critical role in the development and analysis of our perception and planning systems by allowing us to discover and resolve many discrepancies before attempting experiments with the ALV in the field. In addition, it provides the essential link between vehicle control commands and new perceptual inputs. The ultimate objective of these simulations is to close the loop between sensing and acting; that is, between the perception, planning, and vehicle navigation systems. By simulating the terrain, the sensor, and the vehicle, we are able to test the efficiency, correctness, and usefulness of the virtual sensors and behaviors as they are developed.

To provide an accurate model of the ALV terrain, we extract a portion of the five meter resolution map elevation data from ETL and interpolate a smooth surface. Also we allow specific objects such as contours, ramps, plateaus, walls, cliffs, and ravines to be inserted in terms of elevations. The resulting surface defines the basic structure of the underlying ground. We place cultural features such as bushes, ditches, rocks, and grass over the ground surface data. Finally, additive noise is applied to "randomize" the terrain. Interpolation schemes are used to smooth areas or produce gently sloping terrain. The resulting terrain provides both a source of data for the synthetic scanner and the surface on which the simulated vehicle moves.

To simulate the laser range scan, we apply a ray-tracing algorithm in the synthetic terrain to produce a synthetic range scan from any vehicle position and orientation. The synthetic scanner reproduces many of the image artifacts observed in actual scans, including those due to vehicle motion during image acquisition. Using the simulation, we are able to analyze the effect of the sensor depression angle upon obstacle detection.

The vehicle simulation includes parameters for vehicle dimension,the vehicle sensors for location and orientation, and vehicle dynamics such as acceleration and braking. We can simulate collision by applying the vehicle model. In addition, we have simulated the actual MMAC/ALV control algorithm. This was necessary to test the conversion of the control algorithms based on speed and turn-rate control used by Hughes into the vector control algorithms used at MMAC. With this simulation, we can evaluate the effects of time delays between command and vehicle action.

The interface simulation also includes the ability to mimic the software and hardware configuration at MMAC. The full simulation requires the use of several Symbolics Lisp machines and Sun workstations joined via pronet interfaces. This environment identifies and resolves many inconsistencies and shortcomings in the system interfaces and load distribution without consuming valuable time at the test site. Most importantly, such preparation allows us to perform meaningful and successful cross-country experiments within a relatively short experimentation period at MMAC.

### 1.6 Experimentation Results

The experiments run on the ALV were designed to determine the feasibility of cross-country terrain navigation and to demonstrate the Hughes perception and planning software. The experiments accomplished both objectives [7].

Vehicle Configuration

The cross-country experiments required the integration of multiple computers distributed on-board the vehicle and in the ALV lab as shown in Figure 5. The perception system resided on two Sun workstations on-board the vehicle; one Sun was used for the virtual sensors, the other was used to archive data and experiment records. The planning system resided on two Lisp machines in the ALV lab: one Symbolics ran the reflexive behaviors, the other was employed for map-based planning. In addition, vehicle control was interfaced (by MMAC) through an Intel computer.

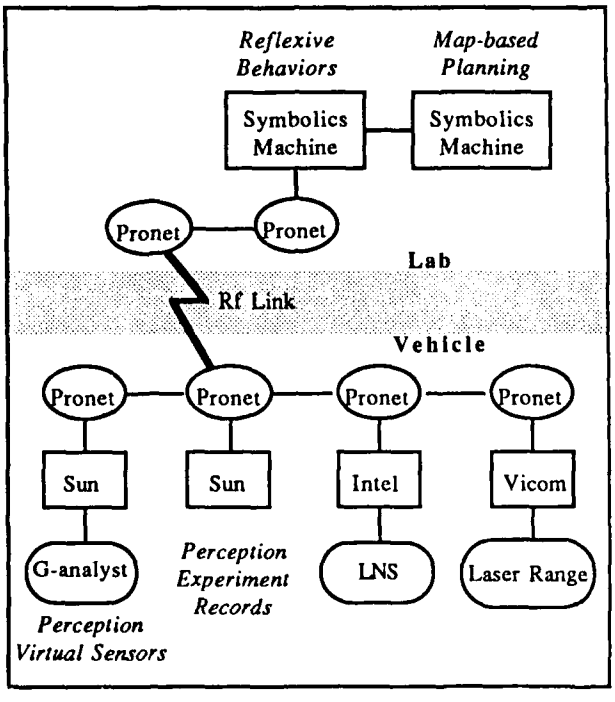



**Figure 5.**
**Experimentation System Configuration**

Perception Summary for Cross-country Navigation Experiments

For the cross-country experiments, we chose a site rich with obstacles and potential paths for obstacle avoidance. The area is a hillside consisting of steep slopes (some over 15 degrees), rock outcrops, large scrub oaks, very small junipers (15 inches high), as well as a narrow sign post (post approximately 2 inches wide and sign approximately 4 inches by 6 inches). In addition, the area includes a complex set of gullies and ravines caused by rain and snow runoff. These gullies span a range of widths from 6 to 25 inches and depths from 4 to 30 inches. A narrow area with a very constrained approach angle is available to cross the area of gullies. The soil is loose and sandy in the vicinity of the gullies resulting in mud and vehicle slippage during the December experiments. We also set up an obstacle course on a flat field at the far end of our experimentation area. The difficulties associated with this test area are sufficient to require caution by a human driver.

In general, the perception system adequately perceived the environment in that the obstacles were detected. The VMT virtual sensor was used with seven trajectories applied on every other pixel in a CEM with 1 foot per pixel resolution. We experienced the greatest difficulty with gullies and rocks "hidden" in the grass. The planning system utilized the VMT information to successfully navigate around obstacles; the average speed for the recent experiments was 3 km/hour. Specific experiments and paths are described in greater detail in another paper in these proceedings [2].

Cross-country Navigation Conclusions

Autonomous cross-country navigation with obstacle avoidance was successfully demonstrated. It is the first such demonstration which integrated map-based and sensor-based vehicle control. It also demonstrated the feasibility of an experimental system operating with conventional computing hardware located both on-board the ALV and remotely in the ALV laboratory. The concepts of behaviors and virtual sensors have been shown to provide a viable approach to local vehicle control, responding reliably to the dynamic conditions of the real world. Lastly, it is significant that Hughes was able to accomplish so much with relatively little vehicle time (with only 2 weeks for the first experiment and 1 week for the second). Much of this success must be attributed to the preparation at Hughes through extensive simulation. These experiments represent significant technological progress for autonomous vehicles.

## 2.0 Multiresolution Object Detection

### 2.1 Problem

The interpretation of large high resolution images with highly variable backgrounds can be facilitated by examining features extracted from multiple resolutions of the image. The objects of interest are modeled at each resolution in terms of features that can be used to provide evidence for the objects. By examining lower resolutions during the initial stages of image interpretation, object hypotheses can be made based on large, prominent features. Given these initial hypotheses, higher

resolutions are examined only in those areas in which objects of interest are expected.

### 2.2 Approach

An object is modeled according to its expected characteristics in the image using two kinds of features: salient features that create initial object hypotheses, and supporting features that provide evidence for the hypothesized object. At each resolution, hypotheses are generated in two ways: first, a hypothesis may result from a feature that is due to an instance of the object, and second, a hypothesis may result from objects that already exists at either the same or a lower resolution. In either case, the object creates hypotheses in accordance with the model that specifies the confirming evidence.

The image interpretation system does not follow an algorithmic approach, but instead chooses procedures based on the current state. The approach is both data-driven and model-driven, utilizes hypothesis generation and verification, and employs evidential reasoning to evaluate the hypotheses. The system adheres to the principle of least commitment in two ways: 1) object hypotheses occur only if there exists supporting intrinsic feature properties, and 2) final interpretations are not determined until all hypotheses have been made.

### 2.3 Preliminary results

The system has been exercised on two aerial images: one consisting of a single submarine and the other consisting of three airplanes. The submarine scenario had sufficient resolution to analyze using three resolutions. Features at each resolution that were used as evidence for a submarine are shown in Figure 6. The airplane image was analyzed using only two resolutions. Features for detection of an airplane are shown in Figure 7.

In both of these examples, the original object hypotheses are made at the higher levels. These hypotheses are then projected to the lower levels. In the submarine example, the lowest level model contains additional detail, namely, the tail. In the airplane image, only those areas near the original hypotheses are considered at the lower level, thereby making the interpretation process more efficient. The features used to extract the airplanes are independent of the features used for the submarines. By appropriately choosing features the interpretation of a poor quality image was possible. A more complete system description and discussion of results is found elsewhere in these proceedings [3].

***Acknowledgements***. The progress presented in this overview represents the combined efforts of the technical staff members of the Computer Vision and Autonomous Planning Sections at the Hughes AI Centers. In addition we acknowledge the many valuable discussions with Dr. R. Nevatia. We also thank D.Y. Tseng for his continued support of our technical pursuits.

## References

[1] M.J. Daily, J.G. Harris, K. Reiser, "An Operational Perception System for Cross Country Navigation", Proceedings Image Understanding Workshop, Boston, MA, April, 1988.

[2] D.M. Keirsey, D.W. Payton, J.K. Rosenblatt, "Autonomous Navigation in Cross Country Terrain", Proceedings Image Understanding Workshop, Boston, MA, April, 1988.

[3] T.M. Silberberg, "Multiresolution Aerial Image Interpretation", Proceedings Image Understanding Workshop, Boston, MA, April, 1988.

[4] K.E. Olin, F.M. Vilnrotter, M.J. Daily, K. Reiser, "Developments in Knowledge-Based Vision for Obstacle Detection and Avoidance", Proceedings Image Understanding Workshop, Los Angeles, CA, February, 1987.

[5] D.W. Payton, "An Architecture for Reflexive Control", Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA., 1986.

[6] R.S. Wallace, M.D. Howard, "HBA Vision Architecture: Built and Benchmarked", Proceedings Workshop on Computer Architecture for Pattern Analysis and Machine Intelligence, Seattle, WA, October, 1987.

[7] M. Daily, J. Harris, D. Keirsey, K. Olin, D. Payton, K. Reiser, J. Rosenblatt, D. Tseng, and V. Wong, "Autonomous Cross-Country Navigation with the ALV", DARPA Planning Workshop, Texas, December, 1987.

| Object | Level | Feature | Hypothesis |
|---|---|---|---|
| Submarine | 0,1,2 | Parallel Line Pair | Initial |
| Submarine Area | 0,1,2 | Region | Confirm |
| Shadow/ Sail | 0,1,2 | Region | Confirm |
| Glint | 1,2 | Line | Confirm |
| Tail | 0 | Region | Confirm |

**Figure 6. Submarine Model**

| Object | Level | Feature | Hypothesis |
|---|---|---|---|
| Airplane | 0,1 | "Fuselage" | |
| Fuselage | 0,1 | Region | Initial |
| Wing | 0,1 | Region | Confirm |
| Shadow | 0,1 | Region | Confirm |

**Figure 7. Airplane Model**

# Image Understanding Research at GE [1]

J.L. Mundy

General Electric Corporate Research and Development Center
Schenectady, NY 12301

**Abstract**

The major goals and research results for the Image Understanding Project at the GE Corporate Research and Development Center are reviewed. The primary reasearch emphasis is on model based object recognition and geometric reasoning.

## 1 Model-Based Object Recognition

The use of explicit geometric models has proved to be an effective approach to recognize and locate objects in cluttered natural scenes. The geometric constraints imposed by a three-dimensional model are able to eliminate false matches to background and clutter features. In addition, the process of matching the model to image data produces the coordinate transformation between the model and the image reference frame.

One major goal of the project is to demonstrate the effectiveness of model-based image understanding in the context of military reconnaissance and target recognition applications. The first year of effort has resulted in a successful set of experiments which apply a three-dimensional matching system to the recognition of aircraft and vehicles in aerial views.

These experiments have shown that the constraints provided by pairs of vertices and associated edges are sufficient to determine the correct match between an object model and an unsupervised segmentation of the image data into two-dimensional edges and vertices [Thompson and Mundy 87a]. These concepts can also be extended to tracking moving objects in three-dimensional space, over time.[Thompson and Mundy 87b].

We also have demonstrated that the matching algorithm can be implemented effectively on a fine-grained SIMD architecture, the Connection Machine$^{TM}$[Thompson and Mundy 87c]. Some preliminary results indicate that a Connection Machine(CM1) with 16K processors is able to execute the algorithm around 100X faster than the Symbolics Lisp Machine. We are now reimplementing the algorithm on a more recent version of the Connection Machine(CM2) to make more extensive timing comparisons.

A recent theme in our matching research is the analysis of error in the determination of the transformation between the model and the image coordinate frames. We have developed the concept of *effective viewing* that provides a measure of the transformation accuracy of a given vertex-pair feature, as a function of viewpoint [Mundy et al]. This error measure will permit the automation of model feature selection and provide a weighting function for features in clustering.

[1]This work was supported in part by the DARPA Strategic Computing Vision Program in conjunction with the Army Engineer Topographic Laboratories under Contract No. DACA76-86-C-0007.

## 2 Geometric Model Construction

A second major goal of the project is to develop methods for automatically generating models. The primary emphasis is on models for object matching. However, the rapidly expanding applications of image simulation are also creating a demand for extensive libraries of three-dimensional solid models. In fact, the two applications are likely to become closely integrated, where simulated displays are generated of tactical and strategic sites, based on automatic recognition of objects from aerial views of the sites.

### 2.1 View Intersection

Two methods for the generation of models are under investigation. The first method involves the use of a range sensor which uses triangulation (or time of flight) to determine a set of three-dimensional coordinates on the surface of an object. These point samples are grouped into a partial, polyhedral surface description of the visible object surface. This partial set of object faces is projected along the view projection axes to form a closed, view solid.

A number of such polyhedra, taken from a number of viewpoints are intersected to obtain an estimate of the total object boundary[Connolly and Stenstrom]. The general approach has also been used to extract an object model from a set of intensity views. In this case, the view solids are generated by extracting occlusion boundaries from the intensity image data. In this case, only convex objects can be completely reconstructed.

This method has already been successful in generating a model for a polyhedral object which is suitable for recognizing the object from any viewpoint[Connolly et al].

### 2.2 Symbolic Models

A second method involves the use of geometric reasoning techniques to establish a set of constraints for an object as seen in a single two-dimensional projection. In this approach the object is represented as a set of algebraic constraints. The effort so far has been to develop techniques for efficiently reasoning about geometric relationships and to demonstrate these techniques in solving realistic geometry problems.

Two major accomplishments have been achieved during the first year in this line of development. A geometric reasoning system, which uses algebraic deduction to prove geometry theorems has been developed. The system, called GEometer, has been able to prove hundreds of geometry theorems. GEometer is implemented on the Symbolics Lisp Machine and uses a deduction system called the Groebner basis to prove theorems.

GEometer also has a powerful graphical user interface to enter geometric axioms and constraints [Cyrluk et al 87].

In the second geometric reasoning development, the Groebner basis method was extended to prove the consistency between two views of a three-dimensional object. In these experiments, an ideal two-dimensional view of a polyhedra is used to generate a system of algebraic constraints. These constraints are specified in terms of unknown depths of the object vertices. These depths cannot be determined from a single two-dimensional view. Instead, the two-dimensional projection is used to generate a system of equations with the unknown vertex depth coordinates taken as variables. This system of equations can then be used as a model to determine whether or not a second two-dimensional view is consistent with some projection of the object in the first view [Cyrluk et al 87].

The consistency of the two views can be established with the same proof methods used in GEometer. In this approach, the first two-dimensional image is being used as a partially determined three-dimensional model. The experiments were able to demonstrate the validity of this approach for ideal projections, i.e. projections in which the vertex coordinates are given exactly. The major demonstration is that relatively few constraints are sufficient to prove the inconsistency between two views. In the case that the views are consistent, it is then possible to determine the transformation between views and extend the model to include explicit three-dimensional constraints.

This technique has been extended to include the ability to reason about inequalities. These inequalities arise due to uncertainty in the projection coordinates and in the object model constraints. We have integrated the Groebner basis technique with the SUP-INF method to provide a unified approach to reasoning about empirical geometric relations[Cyrluk et al 88].

## 3 PACE

We have initiated a new project this year to provide a practical application focus for the research activities just described. The project is to develop an integrated environment for intelligence analysis, called the Perceptual Analysis and Control Environment, or PACE.

The goal of PACE is to use detailed object models and a terrain description to determine the status of a military site, such as an airbase. We will also incorporate constraints on object location and orientation that are derived from natural language reports about the site. These constraints are used to focus the search during object recognition. The resulting world description is displayed in synthetic form using reflectance image mapping. This user interface allows the intelligence analyst to interact with the object recognition and object modelling tools.

We are developing PACE in cooperation with SRI. In particular, we will make use of The Cartographic Workstation for object modeling and display as well as CKS for communicating between different conceptual representations. We have selected the Schenectady 109th Tactical Airlift Group to provide experimental image data and actual mission message traffic[Corby et al].

## References


[Thompson and Mundy 87a] Thompson, D., Mundy, J.L., "Three-Dimensional Model Matching From an Unconstrained Viewpoint", Proc. IEEE Robotics and Automation, 1987, pp. 280.

[Thompson and Mundy 87c]
Thompson,D., Mundy, J.L.,"Model-Directed Object Recognition on the Connection Machine,"Proc. DARPA Image Understanding Workshop, 1987.

[Thompson and Mundy 87b] Thompson,D.,
Mundy, J.L.,"Model-Based Motion Analysis,"Proc. 4th International Symposium on Robotics Research, MIT Press, 1988.

[Mundy et al] "The Concept of an Effective Viewpoint," Mundy, J.L.,Heller, A.J., and Thompson, D.W., Proc. DARPA Image Understanding Workshop, Morgan Kaufmann, Los Altos,Ca., 1988.

[Connolly and Stenstrom] Connolly,C.I.
and Stenstrom, J.R.,"Generation of Face-Edge-Vertex Models Directly From Images,"Proc. DARPA Image Understanding Workshop, 1988.

[Connolly et al] Connoly,C.I.,Mundy,J.L.,Stenstrom,J.R.,
Thompson,D.W.,"Matching From Three Dimensional Range Models Into 2-D Intensity Scenes,"First International Conference on Computer Vision, June 1987,p.73.

[Cyrluk et al 87] Cyrluk,D., Kapur,D.,Mundy,J.L., Nguyen, V.,"The Formation of Partial 3D Models From 2D Projections - An Application of Algebraic Reasoning,"Proc. DARPA Image Understanding Workshop,1987.

[Cyrluk et al 88] Cyrluk,D., Kapur,D.,Mundy,J.L.,"Algebraic Reasoning in View Consistency and Parameterized Model Matching Problems,"Proc. DARPA Image Understanding Workshop,1988

[Corby et al] Corby, N.,Mundy,J.L. and Vrobel,P., "PACE-An Environment For Intellgence Analysis,"Proc. DARPA Image Understanding Workshop, 1988.

# Qualitative Reasoning and Modeling for Robust Target Tracking and Recognition from a Mobile Platform

**Bir Bhanu and Durga Panda**

Honeywell Systems & Research Center
3660 Technology Drive, Minneapolis, MN 55418

## ABSTRACT

In the DARPA Strategic Computing Computer Vision Program, we focus on demonstrating robust techniques for target tracking and recognition from a moving robotic combat vehicle. Our work is specifically directed towards significant enhancements in the performance of existing target tracking techniques under high clutter and low contrast situations in a ground-to-ground scenario when the robotic combat vehicle is in motion and multiple targets may appear at varying ranges. The topics currently under investigation are: decomposition of complex vehicle motion into its constituent parts; qualitative 3-D scene modeling; target motion detection and tracking; landmark recognition; 3-D target model acquisition and refinement; and use of recognition and map information in an integrated motion detection and tracking system. The results from our research are useful in vision controlled navigation/guidance of a robotic combat vehicle for practical military missions such as targeting, reconnaissance and surveillance. This report summarizes the progress made during the period from March 1987 to January 1988. We also discuss the technology transfer aspects of our work.

## 1. INTRODUCTION

The goal of our research in the Strategic Computing Computer Vision Program is to demonstrate robust techniques for target tracking and recognition from an autonomously-moving robotic combat vehicle. In our experience in implementing vision controlled navigation/guidance for reconnaissance, surveillance, search and rescue, and targeting missions, we find that for spatio-temporal vision problems, purely quantitative approaches are unsuitable and insufficient because of the inexact nature of vision. As such, the technical basis of our work is qualitative reasoning and modeling for dynamic scene understanding.

To achieve our goal, we are engaged in developing efficient and reliable techniques for qualitative motion understanding, dynamic model matching, automatic 3-D model acquisition, spatial reasoning, geographic knowledge representation and its use in recognition and tracking. This work is specifically directed towards significant enhancements in the performance of existing target tracking techniques under high clutter and low contrast situations in a ground-to-ground scenario when the robotic combat vehicle is in motion and multiple targets may appear at varying ranges.

### 1.1 Qualitative Reasoning and Modeling

The choice of a suitable scheme for representing the perceived state of the scene, observed by a moving robotic combat vehicle, is a crucial question. It has an immediate impact upon the efficiency, versatility, and robustness of the reasoning processes that are attached to this representation. It is questionable whether an accurate numerical description of the 3-D environment is really necessary to facilitate efficient reasoning of spatio-temporal processes. The use of *qualitative* descriptions of physical properties has raised considerable interest in the area of Artificial Intelligence.[16,21] Its potential significance to the field of computer vision has been addressed only recently.[4,38,41] The main argument is that many of the error-prone, computationally expensive techniques which are commonly used can be replaced by emphasizing the qualitative effects and utilizing less precise representations without sacrificing the usefulness of the results.

Most previous work in motion understanding attempted to obtain the 3-D scene structure from motion in the form of a quantitative, numerical description of the spatial layout of the environment relative to the camera. The problems related to this approach are well-known and applications using real imagery have been rare. The systems of nonlinear equations that must be solved for this purpose are numerically unstable; small errors in the estimate of image displacement lead to unproportionally large errors in the estimated 3-D geometry.

Since numerical schemes are designed to converge towards a single solution which is optimal in some sense, there seems to be no practical mechanism that would reflect the *uncertainty* of the input data on the final result. Furthermore, the necessary assumption of rigidness cannot be guaranteed. When features are assumed to form a rigid configuration in space but are actually moving relative to each other, this may still result in a rigid interpretation. The problem with this approach is how the numerical model responds when moving features and stationary features are inadvertently grouped. In the best case, the deviation from a rigid configuration would be indicated by a high error value for the feature which is actually in motion. If this is not the case, the model may converge towards a completely different

solution.

Following the qualitative reasoning and modeling approach, the central building block of our DRIVE system[4] for target motion detection and tracking is a *Qualitative Scene Model* (QSM), which can be considered as the "mind" of the motion understanding system. This model is a 3-D, camera-centered representation of the scene which describes the observed environment by using a set of simple qualitative relationships. The set of entities in the QSM is conceptually split into two parts, the *stationary world* and a set of independently moving objects. Construction of the QSM over time is accomplished by a reasoning process which draws conclusions from significant configurations and changes in the image. As the vehicle travels through the environment, the model is continuously updated and revised by adding or deleting hypotheses.

Additionally, the state of the QSM is a not a single interpretation but a *set* of interpretations which are all pursued simultaneously. This provides a very flexible mechanism for handling the inherent ambiguities encountered in image understanding. Each interpretation is a collection of hypotheses, called *partial* interpretations, which cover overlapping subsets of the entities in the model. The structure and dynamic behavior of the Qualitative Scene Model are described in more detail in the paper by Bhanu and Burger.[6]

Qualitative reasoning and modeling is also emphasized in our work on landmark and target recognition from a mobile platform.[7,29] Using qualitative information, we do not have to rely on obtaining precise geometric representations of a target. To handle continuous changes in the target's appearance caused by range and perspective, we use a dynamic model matching technique,[29] which combines a camera model, 3-D target models, and predicted range and perspective to generate multiple 2-D image models for matching. TRIPLE's[7] machine learning approach allows for automated 3-D model acquisition and refinement. It uses qualitative and quantitative shape descriptions.

The research results described in this report are partitioned into the following topic areas: (a) target motion detection and tracking and (b) landmark and target recognition. We also discuss the technology transfer aspects of our application in the discussion.

## 2. TARGET MOTION DETECTION AND TRACKING

Motion becomes a natural component of visual information processing as soon as moving objects are encountered in some form; while following a convoy, approaching other vehicles, or detecting threats. The presence of moving objects and their behavior must be known to provide appropriate counteraction. In addition, image motion provides important clues about the spatial layout of the environment and about the actual movements of the vehicle. As part of the vehicle control loop, visual motion feedback is essential for path stabilization, steering, and braking. Results from psychophysics[24,34] show that humans rely heavily on visual motion for motor control.

While the vehicle is moving itself, the entire camera image is changing continuously, even if the observed part of the environment is completely stationary. The interpretation of complex dynamic scenes is therefore the continuous task for the vision system of an autonomous robotic combat vehicle. Previous work in motion analysis has mainly concentrated on numerical approaches for the reconstruction of motion and scene structure from image sequences. Recently Nagel[28] has given a comprehensive review. While a completely stationary environment has been assumed in most previous work on the reconstruction of *camera motion*, the possible presence of moving objects must be accounted for in this scenario. Similarly, one cannot rely on a fixed camera setup to *detect* those moving objects. Clearly, some kind of common reference is required against which the movement of the vehicle as well as the movement of objects in the scene can be related.

Extensive work has been done in determining the relative motion and rigid 3-D structure from a set of image points and their displacements, basically following two approaches.

In the first approach, 3-D structure and motion are computed in one integral step by solving a system of linear or nonlinear equations[27,39] from a minimum number of points on a rigid object. The method is reportedly sensitive to noise.[15,42] Recent work[10,11,17,37,40] has addressed the problem of recovering and refining 3-D structure from motion over extended periods of time, demonstrating that fairly robust results can be obtained. However, these approaches require large amounts of computation, convergence is slow and require many distinct views of the object (the environment), which are generally not available to a moving vehicle. In addition, it seems that the noise problem cannot be overcome by simply increasing the time of observation.

The second approach[10,18,24,25,33,35] makes use of the unique expansion pattern which is experienced by a moving observer. Arbitrary observer motion can be decomposed into translational and rotational components from the 2-D image without computing the structure of the scene. In the case of pure camera translation in a stationary environment, every point in the image seems to expand from one particular image location termed the *Focus of Expansion* (FOE). The closer a point is in 3-D, the more rapidly its image expands away from the FOE. Thus, for a stationary scene, the 3-D structure can be obtained directly from the expansion pattern. Certain forms of 3-D motion become apparent by local deviations from the expanding displacement field and therefore can be detected immediately. The views of the scene need not be very distinct in this approach and there seems to be evidence from psychophysics that the human visual system employs similar techniques.[32,34]

The primary goal for *Dynamic Scene Understanding* in this particular context is to construct and maintain consistent and plausible interpretations of the time-varying images obtained from the camera on the moving vehicle by determining:

- How is the vehicle itself moving ?
- What is the approximate 3-D structure of the scene ?
- What is moving in the scene and how does it move ?

Obviously, these three goals are in very close interaction: any form of motion, whether vehicle motion or actual target motion, must be measured against some stationary reference in the environment.

We have developed a new DRIVE (Dynamic Reasoning from Integrated Visual Evidence) approach based on a *Qualitative Scene Model* to solve the motion understanding problem. The approach addresses the key problems of the estimation of vehicle motion from visual cues, the detection and tracking of moving objects, and the construction and maintenance of a global dynamic reference model. Object recognition, world knowledge, and accumulation of evidence over time are used to disambiguate the situation and continuously refine the global reference model. The approach departs from previous work by emphasizing a qualitative line of reasoning [16,21] and modeling, where multiple interpretations of the scene are pursued simultaneously in a hypothesis and test paradigm. Different sources of visual information such as two-dimensional displacement field, spatial reasoning, and semantics are integrated in a rule-based framework to construct and maintain a vehicle centered three-dimensional model of the scene. This approach offers significant advantages over "hard" numerical techniques which have been proposed in the motion understanding literature.[26,36] These advantages include the tracking of objects in the presence of partial or total occlusion and use of this information for route planning and threat handling.

Details of the qualitative reasoning concept emphasizing the motion aspects of the DRIVE system are presented in papers by Bhanu and Burger.[4,5,6,12,13,14]

### 2.1 Estimation of Vehicle Motion

The problem of determining the motion parameters of a moving camera relative to its environment from a sequence of images is crucial for the application of computer vision to practical military missions. In addition to translating in an unknown direction, the vehicle also rotates about an arbitrary axis (roll, pitch, and yaw), though not drastically. However, due to the design of the vehicle, the direction of travel is quite restricted, e.g., vehicle orientation does not change rapidly and target stays within the field of view. The observed displacement field is the addition of the vector fields caused by vehicle translation and rotation, such that the vehicle motion cannot be obtained from the displacement field directly. However, the displacement field caused by the vehicle's motion can be decomposed into its rotational and translational components exclusively in the 2-D image, without any 3-D information.

In our work the computation of camera motion is performed from sets of displacement vectors obtained from consecutive pairs of images.[19] First, the decomposition of 3-D camera motion into rotation and translation components and their individual effects upon the image are analyzed in detail. Two basic approaches for computing camera motion are evaluated. In the *FOE-from-Rotation* approach, the direction of camera translation (marked by the *Focus of Expansion* - FOE) is derived for a given estimate of the camera's rotation. Alternatively, in the *Rotation-from-FOE* approach, the rotational components are determined from a given estimate of the location of the FOE. It is shown that the latter approach is highly robust against disturbances of the displacement field, since it works without extending the displacement vectors. Instead of searching for one particular FOE, the final algorithm computes a connected *region* of possible FOE locations, which accounts for noise and distortions in the image. Finally, the absolute velocity of the vehicle towards the FOE is estimated from the expansion pattern by knowing the height of the camera above the (approximately flat) ground. We show the results on real image sequences in the paper by Bhanu and Burger.[6]

### 2.2 Estimation of Stationary 3-D Structure

The environment is modeled as a 3-D, time-varying configuration of rigid objects whose structures, relative positions, and motions are estimated from visual information. The stationary part of the world is represented by a subset of the rigid objects, which form a rigid configuration in 3-D space. This definition, however, is not sufficient to identify the stationary world a priori, because more than one rigid subset of world objects may be observed. To operate in a real environment, some description about the 3-D layout of the scene must be available. In the DRIVE approach, a vehicle-centered model of the scene is constructed and maintained over time, representing the current set of feasible interpretations of the scene. In contrast to most previous approaches, no attempt is made to obtain an accurate geometric description of the scene. Instead, a *Qualitative Scene Model* is proposed which holds only a coarse qualitative representation of the three-dimensional environment. As part of this model, the "stationary world" is represented by a set of image locations, forming a rigid 3-D configuration which is believed to be stationary. All the motion-related processes at the intermediate level of vision use this model as a central reference. The motion of the vehicle, for instance, is related to the stationary parts of the environment, even if large parts of the image are in motion. This kind of reasoning and modeling appears to be sufficient and effective for this problem.

### 2.3 Detection of Moving Targets

For intelligent action in the presence of potential threats and targets, navigation in a traffic environment, etc., information on actual motion in the scene is indispensable. Moving objects must be detected and isolated from the stationary environment, their current motions must be estimated to track them, and expectations about their future behavior must be created. Since the camera itself is moving, the stationary part of the scene cannot be assumed to be registered in subsequent images, as in the case of a stationary sensor. Simple frame-differencing techniques to detect and isolate moving objects do not work in this case because image changes, due to sensor motion, would generate too many

false alarms. More sophisticated image-based techniques, which apply 2-D transformations (warping) to the image to compensate for background motion, work well only when objects are moving in front of a relatively flat background, such as in some air-to-ground applications. To detect actual object motion in the complex scenario of a robotic combat vehicle, the 3-D structure of the observed environment together, with the vehicle's motion, must be taken into account.

In our DRIVE approach, 3-D motion is detected in two ways:

- *First*, some forms of motion are concluded directly from the 2-D displacement vectors without any knowledge about the underlying 3-D structure.
- *Second*, motion is detected by discovering inconsistencies between the current state of the internal 3-D scene model and the changes actually observed in the image.

### 2.4 Interpretation of Terrain

An autonomous robotic combat vehicle must be able to navigate not only on the roads, but also through terrain in order to execute its missions of surveillance, search and rescue, and munitions deployment. To do this the vehicle must categorize the terrain regions it encounters as to the trafficability of the regions, the land cover of the regions, and region-to-map correspondence. Our approach for terrain interpretation employs a robust texture-based algorithm and a hierarchical region labeling scheme for ERIM 12 channel Multi-Spectral Scanner data. The technique, called HSGM (Hierarchical Symbolic Grouping for Multi-spectral data), is specifically designed for multi-spectral imagery, but is appropriate for other categories of imagery as well. For this approach, features used for segmentation vary from macro-scale features at the first level of the hierarchy to micro-scale features at the lowest level. Examples of labels at the macro-level are sky, forest, field, mountain, road, etc. For each succeeding level of the hierarchy, the identified regions from the previous stage are further subdivided, if appropriate, and each region's labeling is made more precise. The process continues until the last stage is reached and no further subdivision of regions from the preceding stage appears to be necessary. Examples of region labels for this level of the hierarchy are gravel road, snowberry shrub, gambel oak tree, rocky ledge, etc.

Details of the HSGM technique with results and examples from real imagery are given in papers by Bhanu and Symosek.[8,9]

### 2.5 Map Integrated Motion Detection and Tracking

A priori information for scene content, in the form of digital map data, is an invaluable resource for tracking algorithms. Contextual information, derivable from digital maps, is especially critical to high-level reasoning paradigms which carry out the mission tasks such as estimation of vehicle location, condition monitoring, target acquisition, target classification, target tracking, target engagement, and sensing of vehicle orientation.

Using techniques developed at Honeywell, digital map databases can be transformed into digital visibility maps, from which intervisibility predictions can be computed.[20] We are in the process of implementing a map reasoning system that will be able to identify the world position of moving and tracked targets. The system will incorporate map/terrain and cartographic data bases and will be integrated with the DRIVE system. DRIVE will select moving targets in the image, give their 2-D image location, velocity vector, and approximate range. Given this information from DRIVE, the vehicle geodetic location, and a camera model, the system will establish an image-to-map registration and search in the map data base for possible roads/terrain on which the targets may be moving. The system will also provide information about neighboring landmarks to the target and possible occlusion information.

## 3. Landmark and Target Recognition

A few of the desirable features to be incorporated in an advanced target recognition system[3] are: (a) The models used by the system to represent targets, contexts, and other system knowledge should be dynamic data structures; (b) Most data should be of a symbolic, qualitative nature, thus avoiding the problems encountered in dealing with quantitative information. Using qualitative information, we do not have to rely on obtaining precise geometric representations of target; (c) The system has to be able to handle problems such as imprecise segmentation, occlusion, noise, etc. and ; (d) The system should exhibit improved performance over time. This improvement may come in the form of faster recognition times, improved recognition accuracy, and higher confidence in system results.

Our work on landmark and target recognition is directed towards emphasizing the above features in a dynamic scenario. Target recognition from a mobile platform requires the ability to recognize targets from varying range and perspectives under changing environmental conditions.

### 3.1 Landmark Recognition

Landmark recognition is used to update the land navigation system which accumulates a significant amount of error after the vehicle traverses long distances, which is typically the case in surveillance and targeting missions. The vision system of the autonomous vehicle is required to recognize the landmarks as the vehicle approaches from the road or terrain.

We have developed an expectation-driven, knowledge-based landmark recognition system, called PREACTE[31] that uses map, and landmark knowledge, spatial reasoning and a novel dynamic model matching technique.[29] PREACTE's mission is to predict and recognize landmarks as the vehicle approaches them from different perspective angles and at varying ranges. Once the landmarks have been recognized, they are associated with specific map coordinates, which are then compared to the land navigation system's readings, and corrections are made. Landmarks of interest include build-

ings, gates, poles, and other man made objects.

Dynamic model matching generates and matches target landmark and map site descriptions dynamically. These descriptions are a collection of spatial, feature, geometric and semantic models. From an approximate range and view angle, and using a priori map information, 3-D landmark models, and the camera model, PREACTE generates predictions about individual landmark locations in the 2-D image. The parameters of all models are a function of range and view angle. As the vehicle approaches the expected landmark, the image content changes, which in turn requires updating the search and match strategies. Landmark recognition, in this framework, has been divided into three stages: detection, recognition, and verification. At far ranges, only "detection" of distinguishing landmark features is possible, whereas at close ranges, recognition and verification are more feasible, since more details of the object are seen. The salient features of the technique are: (a) landmark models are dynamic; (b) different landmarks require different representations and modeling techniques; (c) a single landmark requires hybrid models; and (d) at different ranges, different matching and recognition plans are performed.

Details of the landmark recognition system, PREACTE, together with results on Autonomous Land Vehicle imagery, are given in the papers by Nasr and Bhanu.[29,30,31]

### 3.2 Target Model Acquisition and Refinement

Target recognition systems currently lack the ability to adapt to changing environmental conditions and to modify their behavior based on the context of the situation in which they are operating. In order to be effective in dynamic outdoor scenarios, a robust recognition system should be able to automatically acquire necessary contextual information from the environment. Most target recognition systems lack this capability. Their performance begins to quickly degrade when subjected to the problems of variable lighting conditions, image noise and object occlusion.

Due to recent advances in machine learning technology, some of the problems encountered in the target recognition domain seem to be resolvable. Learning allows an intelligent recognition system to use situation context in order to understand images. This context, as defined in a machine learning scenario, consists of a collected body of background knowledge as well as environmental observations which may impact the processing of the scene.

Machine learning will facilitate two main breakthroughs in the target recognition domain: automatic knowledge base acquisition and continuous knowledge base refinement. The use of learning in the knowledge base construction will save the user from spending the enormous amount of time necessary to derive target models and databases. Knowledge base refinement can then be incorporated to make any necessary changes to improve the performance of the recognition system. These two modifications alone will serve to significantly advance the present abilities of most target recognition applications

To validate the concept of a target recognition system with integrated machine learning capabilities, the paper by Bhanu and Ming[7] presents an overview of a new approach to target recognition. The system currently under implementation is called TRIPLE: *Target Recognition Incorporating Positive Learning Expertise*. The system uses a multi-strategy technique; two powerful learning methodologies are combined with a knowledge-based matching technique to provide robust target recognition. Using dynamic models, TRIPLE can recognize targets present in the database. If necessary, the models can be refined if errors are found in the representation. Additionally, TRIPLE can automatically store a new target model and recall it when that target is encountered again. Finally, since TRIPLE uses qualitative data structures to represent targets, it can overcome problems such as image noise and occlusion.

The two main learning components of the TRIPLE system are Explanation-Based Learning (EBL) and Structured Conceptual Clustering (SCC). Explanation-based learning provides the ability to build a generalized description of a target class using only one example of that class. Structured conceptual clustering allows the recognition system to classify a target based on simple, conceptual descriptions rather than using a predetermined, numerical measure of similarity. While neither method, used separately, would provide substantial benefits to a target recognition system, they can be combined to offer a consolidated technique which employs the best features of each method and is very robust.

## 4. TECHNOLOGY TRANSFER

In this report, we have presented a summary of our work completed during the last twelve months. Our work is directed towards providing key functionalities of target motion detection and tracking, which are needed in autonomous robotic combat vehicle missions of targeting, reconnaissance, and surveillance. In our experience with accomplishing these practical military missions, we find that for spatio-temporal vision problems, purely quantitative approaches are unsuitable and insufficient because of the inexact nature of vision. As such, the technical basis of our work is qualitative reasoning and modeling for dynamic scene understanding. Our PREACTE module for man made landmark recognition and DRIVE module for motion detection and tracking are ready to be transferred and integrated with Carnegie Mellon University's software. We are also working on integrating the PREACTE and DRIVE modules for an end-to-end simulation demonstrating Honeywell's knowledge-based scene dynamics approach for technology transfer to a robotic combat vehicle.

In addition to the robotic combat vehicle applications as discussed above, our interest is also to transfer this technology to other practical military applications. Precision Guided Weapons (PGWs), such as Honeywell's next generation SADARM, are one such application. Conventional technology, such as Automatic Target Recognition (ATR), has come a long way but it needs help.[3] It is clear that for vision technology to succeed in practical smart weapons applications, it must be optimally suited for such multisensor combi-

nations as millimeter wave/infrared,[1,2,23] and $CO_2$ laser (range, reflectance, Doppler and vibration measurements).[22] We are transferring the knowledge-based technology under development here to smart weapons relevant multisensor applications in order to provide significant improvement in performance in diverse scenarios. We are using multisensory and a priori information (map) in a knowledge-based framework to achieve the required performance which is beyond what conventional ATR technology can provide.

## REFERENCES


1. *Dual-Mode Seeker Study, Final Report, Airforce/Army Contract No. F08635-85-C-0156, Honeywell Systems & Research Center.* May 1986.
2. W. Au, S. Mader, and R. Whillock, "Scene Analysis," Third Triannual Technical Report, Night Vision & Electro-Optics Lab, Contract No. DAAL01-85-C-0429, Honeywell Systems & Research Center (July 1986).
3. B. Bhanu, "Automatic Target Recognition: State of the Art Survey," *IEEE Transactions on Aerospace & Electronic Systems* **AES-22**(4) pp. 364-379 (July 1986).
4. B. Bhanu and W. Burger, "DRIVE: Dynamic Reasoning from Integrated Visual Evidence," *Proc. DARPA Image Understanding Workshop*, pp. 581-588 Morgan Kaufmann Publishers, (Feb. 1987).
5. B. Bhanu and W. Burger, "Qualitative Reasoning in Dynamic Scene Understanding," *Submitted to Computer Vision, Graphics and Image Processing*, (1987).
6. B. Bhanu and W. Burger, "Qualitative Motion Detection and Tracking of Targets from a Mobile Platform," *Proc. DARPA Image Understanding Workshop*, (April 1988).
7. B. Bhanu and J.C. Ming, "TRIPLE: A Multi-Strategy Machine Learning Approach to Target Recognition," Proc. DARPA Image Understanding Workshop (April 1988).
8. B. Bhanu and P. Symosek, "Interpretation of Terrain Using Hierarchical Symbolic Grouping for Multi-Spectral Images," *Proc. DARPA Image Understanding Workshop*, pp. 466-474 (Feb. 1987).
9. B. Bhanu and P. Symosek, "Interpretation of Terrain Using Multispectral Images," *Submitted to Pattern Recognition*, (1987).
10. S. Bharwani, E. Riseman, and A. Hanson, "Refinement Of Environmental Depth Maps Over Multiple Frames," *Proc. IEEE Workshop on Motion, Kiawah Island Resort*, pp. 73-80 (May 1986).
11. T.J. Broida and R. Chellapa, "Kinematics and Structure of a Rigid Object from a Sequence of Noisy Images," *Proc. IEEE Workshop on Motion, Kiawah Island Resort*, pp. 95-100 (May 1986).
12. W. Burger and B. Bhanu, "Estimating 3-D Egomotion from Perspective Image Sequences," *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1987).
13. W. Burger and B. Bhanu, "Qualitative Motion Understanding," *Proc. Tenth International Joint Conference on Artificial Intelligence, IJCAI-87, Milan, Italy*, Morgan Kaufmann Publishers, (August 1987).
14. W Burger and B. Bhanu, "Qualitative Understanding of Scene Dynamic for Autonomous Mobile Robots," *Submitted to International Journal of Robotics Research*, (1987).
15. J.Q. Fang and T.S. Huang, "Some Experiments on Estimating the 3-D Motion Parameters of a Rigid Body from Two Consecutive Image Frames," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-6**(5) pp. 545-554 (September 1984).
16. K.D. Forbus, "Qualitative Process Theory," *Artificial Intelligence* **24**(1-3) pp. 85-168 (December 1984).
17. E.C. Hildreth and N.M. Grzywacz, "The Incremental Recovery of Structure from Motion: Position vs. Velocity Based Formulations," *Proc. IEEE Workshop on Motion, Kiawah Island Resort*, pp. 137-143 (May 1986).
18. R. Jain, S. L. Bartlett, and N. O'Brien, "Motion Stereo Using Ego-Motion Logarithmic Mapping," Proc. Int. Conf. Computer Vision and Pattern Recognition, pp. 188-193 (1986).
19. J. Kim and B. Bhanu, "Motion Disparity Analysis Using Adaptive Windows," Technical Report 87SRC38, Honeywell Systems & Research Center (June 1987).
20. A. Kramer, H. Funk, and J.A. Hawthorne, "Topological Representation," Proc. 1986 IR&D Initiatives Program, Honeywell Systems & Research Center (June 1987).
21. B. Kuipers, "Qualitative Simulation," *Artificial Intelligence* **29** pp. 298-338 (1986).
22. K. Landman, "Automatic Laser Target Classification," Phase I Interim Report, AFWAL/Avionics Lab Contract No. F33615-84 C-1519, Honeywell Systems & Research Center (April 1985).
23. B. Lee, W. Higgins, and J. Gillberg, "Multisensor Algorithm Development: First - Fourth Quarterly Reports," Night Vision and Electro-Optics Lab, Contract no. DAAL 01-85-C-0443, Honeywell Systems & Research Center (1986).
24. D.N. Lee, "The Optic Flow Field: The Foundation of Vision," *Phil. Trans. R. Soc. Lond. B* **290** pp 169-179 (1980).
25. H.C. Longuet-Higgins and K. Prazdny, "The Interpretation of a Moving Retinal Image," *Proc. R. Soc. Lond. B* **208** pp. 385-397 (1980).

26. G. Medioni and Y Yasumoto, "Robust Estimation of 3-D Motion Parameters from a Sequence of Image Frames Using Regularization," Proc. DARPA Image Understanding Workshop, pp. 117-128 (Dec. 1985).

27. A. Mitiche, S. Seida, and J.K. Aggarwal, "Determining Position and Displacement in Space from Images," Proc. IEEE Conf. on Computer Vision and Pattern Recognition, San Francisco, pp. 504-509 (June 1985).

28. H.-H. Nagel, "Image Sequences - Ten (octal) Years - From Phenomenology towards a Theoretical Foundation," Proc. Intern. Conf. on Pattern Recognition, Paris, pp. 1174-1185 (1986).

29. H. Nasr and B. Bhanu, "Dynamic Model Matching for Target Recognition from a Mobile Platform," *Proc. DARPA Image Understanding Workshop*, (April 1988).

30. H. Nasr and B. Bhanu, "Landmark Recognition for Autonomous Mobile Robots," *Proc. IEEE International Conference on Robotics and Automation*, (April 1988).

31. H. Nasr, B. Bhanu, and S. Schaffer, "Guiding the Autonomous Land Vehicle Using Knowledge-Based Landmark Recognition," *Proc. DARPA Image Understanding Workshop*, pp. 432-439 (Feb. 1987).

32. J.A. Perrone, "Anisotropic Responses to Motion Toward and Away from the Eye," *Perception & Psychophysics* **39**(1) pp. 1-8 (1986).

33. K. Prazdny, "Determining the Instantaneous Direction of Motion from Optical Flow Generated by a Curvilinear Moving Observer," *Computer Graphics and Image Processing* **17** pp. 238-248 (1981).

34. D. Regan, K. Beverly, and M. Cynader, "The Visual Perception of Motion in Depth," *Scientific American*, pp. 136-151 (July 1979).

35. J.H. Rieger, "Information in Optical Flows Induced by Curved Paths of Observation," *J. Opt. Soc. Am.* **73**(3) pp. 339-344 (March 1983).

36. E.M. Riseman and A.R Hanson, "Summary of Progress in Image Understanding at the University of Massachusetts," Proc. DARPA Image Understanding Workshop, pp. 48-60 (Dec. 1985).

37. H. Shariat and K.E. Price, "How to Use More Than Two Frames to Estimate Motion," *Proc. IEEE Workshop on Motion, Kiawah Island Resort*, pp. 119-124 (May 1986).

38. W.B. Thompson and J.K. Kearney, "Inexact Vision," *Proc. IEEE Workshop on Motion, Kiawah Island Resort*, pp. 15-21 (1986).

39. R.Y. Tsai and T.S. Huang, "Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects with Curved Surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-6**(1) pp. 13-27 (January 1984).

40. S. Ullman, "Maximizing Rigidity: The Incremental Recovery of 3-D Structure from Rigid and Rubbery Motion," MIT A.I.Memo No. 721 (June 1983).

41. A. Verri and T. Poggio, "Qualitative Information in the Optical Flow," Proc. DARPA Image Understanding Workshop, Los Angeles, pp. 825-834 (February 1987).

42. Y. Yasumoto and G. Medioni, "Experiments in Estimation of 3-D Motion Parameters from a Sequence of Image Frames," *Proc. Int. Conf. Computer Vision and Pattern Recognition*, pp. 89-94 (1985).

# KNOWLEDGE BASED VISION FOR TERRESTRIAL ROBOTS

Daryl T. Lawton, Tod S. Levitt, and Patrice Gelband

Advanced Decision Systems
201 San Antonio Circle, Suite 286
Mountain View, CA 94040-1289

## 1. INTRODUCTION

The Knowledge Based Vision Project [LLG*86, LLM*87a] is concerned with developing terrain recognition and modeling capabilities for an autonomous land vehicle. For functioning in realistic outdoor environments, we are assuming a vehicle with a laser range finder, controllable cameras, and limited inertial sensing. The range finder is used for mapping and navigating through the immediate environment. The cameras are used for object recognition and recognizing distant landmarks beyond the access of the range sensor. We are assuming the vehicle has realistically limited perceptual and object recognition capabilities. In particular, it will see things that it won't be familiar with and can't recognize, but which can be described as stable visual perceptions. The vehicle will not always be able to recognize the same object as being identical from very different points of view. It will have limited, inexact, and undetailed a prior terrain information generally in the form of labeled grid data. One of the basic functions of the vehicle is to elaborate this terrain map of the environment. Another is to successfully navigate through the environment using landmarks.

The underlying functional architecture is similar to other model based vision systems [HR78, Bro84] with attachments for representing large scale space (figure 1). It consists of three different type of knowledge organized into data bases and related inference processes. The **model data base** contains models for perceptual structures and world objects. Primitive perceptual structures correspond to the types of objects produced by image processing routines and include such things as edges, regions, junctions, matches, and difference images. More complete, stable, and environmentally meaningful perceptual structures are produced by **grouping** processes. These structures correspond to connected contours, occlusion/disocclusion boundaries, repeating patterns, and surfaces. The object models correspond to terrestrial objects such as trees, ridges, bodies of water and relations among them such as attachment and relative position. The **scene data base** is where the interpretation of images from a single location is developed. This can include using multiple cameras or a panoramic view from a single camera. We refer to the developed scene model as a **viewframe** in accordance with the types of spatial representations used in [LLCN87a, LLC*87, LLCN87b]. This describes the direction of hypothesized object instances and perceptual groups relative to an observer. It is important to note that the viewframe can consist of either object instance or perceptual group as a basis for using stable, significant visual patterns as landmarks, independent of recognition of them. The **large scale space data base** stores viewframes developed from different locations. Several types of relations between viewframes are stored in the long term data base such as direction of travel between successively extracted viewframes, the relative locations of viewframes, and the objects, such as landmarks, that are common between them.

Two different types of processing flow in this architecture are being studied. The first involves accessing an existing viewframe and using it to recognize predicted objects. In earlier work [LLM*87b] we showed how to develop predicted scene models from a prior terrain information by creating a viewframe from grid data and using it to direct grouping processes to find structures such as predicted road regions, horizon lines, a terrain patch discontinuities. The second involves developing a viewframe without any aprior information. Our current work is based upon a generic terrestrial viewframe model which includes several constraints on the formation of perceptual groups based upon the relative direction of gravity, the horizon line determined by the orientation to the immediate ground plane, and the projected egocentric directions from the observer on this plane. The critical, and as yet unreliable, assumption is that the grouping processes can determine occlusion/disocclusion contours and track groups over time to determine their stability as potential landmarks.

## 2. LONG TERM SPATIAL REPRESENTATION

We have developed a multi-level theory of representation of large-scale space based upon the observation of distinctive visual events, i.e. landmarks (see "Qualitative Navigation II" in this proceedings). This forms the basis of the the representations used in the Large scale space data base and the required attributes of scene models.

The representation consists of three different levels (fig-

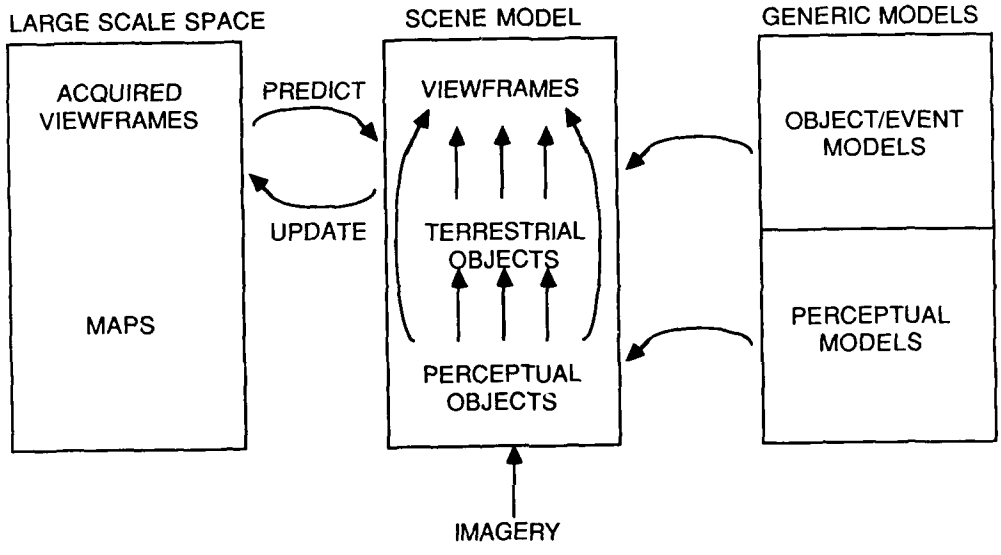



Figure 1: Functional Architecture

A globally consistent metric level in which object and viewer positions are defined with respect to a single coordinate system. This is similar to what is available from DMA/CATTS terrain grids. We have modified our representation of them to support object-level attachments in addition to number-values

- A local metric level consisting of locally attached coordinate systems called viewframes which describe a set of visible landmarks and associated range estimates from a single point of view. Viewframes have a localization area with respect to their landmarks which can be attached to the global terrain grid, other viewframes, and landmarks. In particular, viewframes with overlapping landmark sets can be localized with respect to each other to provide the basis for planning routes between places defined in terms of observable landmarks.

- A topological level consisting of viewframes with no range information. Localization is provided by orientation regions, which are based upon the topological divisions of space using pairs of landmarks.

The representation provides the theoretical foundations for visual memory databases and path planning and execution algorithms that include coordinate free, topological representation of relative spatial location, yet smoothly integrate available metric knowledge of relative or absolute angles and distances. In order to demonstrate our claims, we have built a qualitative navigation simulator, called the QUALNAV model, that provides a software laboratory for experimenting with spatial relationships in visual memory and their relationship to vision-based path planning and execution.

## 3. PERCEPTUAL GROUPING

Perceptual grouping involves structuring images into coherent, stable parts. This is the basis for model matching and extracting environmentally stable information for incorporation into long term memory. Perceptual grouping is necessary for the extraction of significant perceptual events which can correspond to landmarks and for tracking them over time to determine their stability and value as landmarks. It also involves the classification of

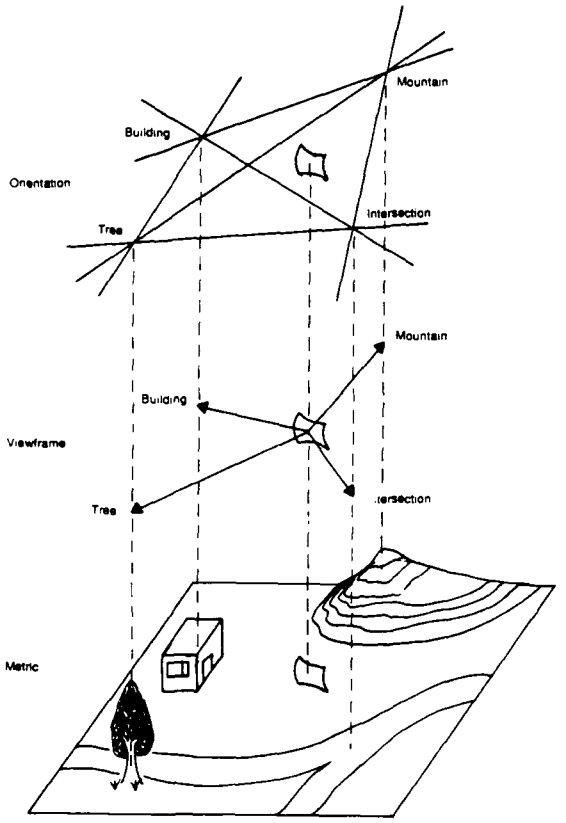



Figure 2: Multi-level Terrain Representation

image structures into a limited number of qualitative types such as occlusion boundaries or surface texture. In previous work [LLM*87b], we developed a grouper based upon parameterized descriptions of contours in terms of image position, color contrast, curvature changes, and other local and global characteristics of a contour. The grouper would then produce an optimization measure from these characteristics that was used in a search process to generate and order consistent contours with these characteristics. This was useful for matching predictions from a prior grid terrain information and for producing larger groups from seed structures which could determine the parameters to initialize the grouper. We are now developing two groupers for determining structural relationships. Both employ the same underlying representation of perceptual objects, but have different control structures. One of them uses an energy minimization formulation. The other has an explicit set of two dimensional patterns used to direct the formation of groups. We are incorporating motion information into the grouping process. There is a natural synergy between these. Grouping processes produce larger image structures which can match more reliably. Motion information can validate potential groups by the common motion of their components.

### 3.1. ENERGY MINIMIZING NETWORK GROUPER

In this grouper, the consistencies and constraints which define perceptual grouping criteria are encoded in the coefficients of an energy functional whose state variables are the perceptual links between image tokens. These links may be either on or off, that is, existent or nonexistent. Minimization of the energy functional results in the formation of perceptual groups of image tokens which are optimal with respect to the grouping constraints.

This work is modeled after Hopfield's neural network approach [Hop84, HT85] to solving non-polynomial optimization problems. The Hopfield approach has been applied with success to problems in early vision [Kea86, KMY86]. Our work represents an extension of parallel network technology beyond the pixel level of vision to intermediate levels which involve symbolic and relational levels of processing.

To accomplish our goal of treating an intermediate level vision problem in a network approach we make use of two key innovations. First, we introduce the concept of 'primitive' symbolic tokens, which is used to manipulate high-level symbolic tokens in a well-defined manner and which play an essential role in the mapping of image data onto the network. By a primitive token we mean one which has only a single reactive site with which it may be linked. For example, if the grouper is performing edge linking, then each half of a given line segment is considered to be a primitive token. The second innovation is the introduction of hierarchical control of the network. This regulates the recursive combination of perceptual groups into large stable groups.

A summary of the manner in which the network grouper operates is the following. A preprocessing stage is used to first extract symbolic tokens from an image or time-sequence of images, and then to reduce these tokens to primitive tokens. A local or global energy minimization procedure is then used to form stable perceptual groups, i.e., linked clusters of tokens. A control process freezes stable groups by forming permanent links from the variable links; it then allows the minimization grouping process to continue at a higher level to produce larger coherent structures. This recursive combination of perceptual groups corresponds to a hierarchy of processing performed by the network. The approach is summarized in Figure 3.

#### 3.1.1. ENERGY FUNCTIONAL

We use a Hopfield energy functional of the form

$$E = -\sum_{ij} T_{ij} l_{ij} - \frac{1}{2} \sum_{ij;kl} T_{ij;kl} l_{ij} l_{kl} + C \sum_{ij} l_{ij}(l_{ij} - 1), \quad (1)$$

where $l_{ij}$ is a boolean variable which is equal to 1 if primitive tokens labeled by $i$ and $j$ are conceptually linked, and is 0 otherwise. The $T$'s can be thought of as lookup tables of real values specifying the compatibility of feature types in a given application. $T_{ij} > 0$ corresponds to the compati-

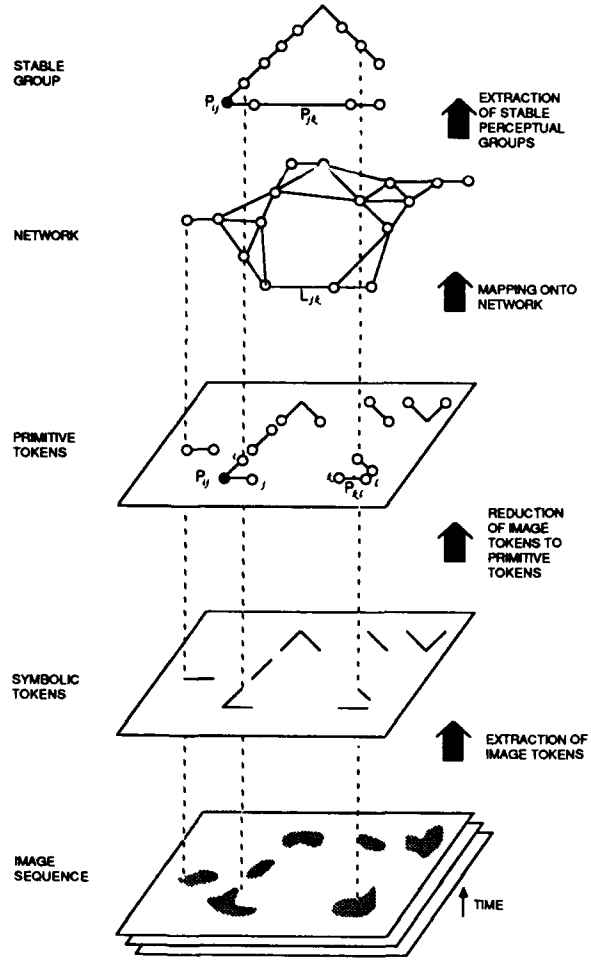



Figure 3: Schematic of our approach (see text). The labels $i, j, \ldots$ refer to 'primitive' tokens, which occupy nodes of the network. The $l_{ij}$ are variable links between primitive tokens; the $P_{ij}$ are permanent links used to bind stable perceptual groups. The recursive (hierarchical) aspects of processing are not shown in this figure.

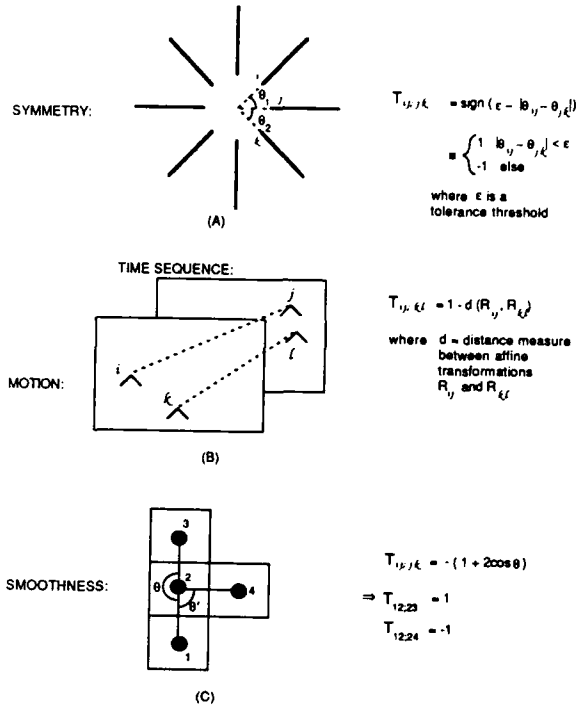



Figure 4: Applications of the second term in the Hopfield energy functional (Eq. (1)). The coefficient $T_{ij;kl}$ compares the relationship of tokens $i$ and $j$ to each other with that of tokens $k$ and $l$. This figure illustrates how this may be used to (a) detect symmetry, (b) track motion, and (c) generate smooth contours.

bility of two features $i$ and $j$. If $T_{ij} > 0$ then minimization of the first term in Eq. (1) yields $l_{ij} = 1$, that is, a link is formed between the tokens. However, if $T_{ij} < 0$, signifying incompatibility, then $l_{ij} = 0$. The comparison of pairs of tokens by this term can be used in such applications as edge linking, region linking, shape matching, and many others.

The second term in Eq. (1) considers the compatibility of *pairs* of conceptual links. Note that if $T_{ij;kl} > 0$, then both $l_{ij}$ and $l_{kl}$ will tend to be equal to 1. However, if $T_{ij;kl} < 0$ so that linking $i$ and $j$ is incompatible with linking $k$ and $l$, then at least one of $l_{ij}$ or $l_{kl}$ must be equal to 0 to minimize this term. Figure 4 illustrates possible applications of this term. (We note that since our knowledge of how visual systems form perceptual groups is imprecise, arbitrariness in the choice of these coefficients is unavoidable.) The last term in Eq. (1), with $C$ a constant chosen sufficiently large, ensures that the $l_{ij}$ will in fact tend toward the values 0 and 1.

We minimize this energy functional using the approach due to Hopfield. First, for each pair $i, j$ we define an internal state variable $u_{ij}$ which is related to $l_{ij}$ through a gain function $g$, which we take to be:

$$l_{ij} = g(u_{ij}) \equiv \frac{1}{1 + \exp(-\lambda u_{ij})}, \qquad (2)$$

where $\lambda$ is a large positive constant. The purpose of this gain function is to restrict $l_{ij}$ to the interval $[0, 1]$. Then, to perform gradient descent on the energy functional, we iterate the differential equation

$$\frac{du_{ij}}{dt} = -u_{ij} + \sum_{ij} T_{ij} + \sum_{ij} T_{ij;kl} l_{ij} - C \sum_{ij} (2l_{ij} - 1), \qquad (3)$$

for each $u_{ij}$. Since the connections $T_{ij;kl}$ in Eq. (1) are symmetric, $E$ is a Lyapunov function and the convergence of this procedure is guaranteed [Hop84].

### 3.1.2. PRIMITIVE TOKENS: MOTIVATION AND DEFINITION

If $i$ and $j$ in Eq. (1) refer to tokens of arbitrary type, then there is no way to refer to specific internal degrees of freedom of these tokens. For example, in an edge-linking application, if $i$ and $j$ refer to line segments, we have no way to specify which ends of the segments we were linking. Because of this ambiguity, it would not be possible, for example, to restrict the formation of unbranched lines, since it would not be possible to differentiate two segments linking to the *same* end of a third segment from two segments linking to opposite ends.

Our solution to this problem is to define the concept of 'primitive' tokens. By a primitive token we mean one which has only a single reactive site with which it may be linked. Links $l_{ij}$ may not be formed to multiple sites within a primitive token. Scalar tokens, that is, tokens which are described by a single value, are by definition primitive. Pixels are examples of scalar tokens, since they are described solely by their grey-scale values.

### 3.1.3. APPROACH TO HIERARCHICAL CONTROL

Much of the formal structure of our approach has been created out of consideration for the intrinsically recursive or hierarchical nature of perceptual grouping. One of the major research issues which remains is the determination of the properties of the controls which are used to go from one level of the hierarchy to another. (We note that we do not intend to imply that these 'levels' are well-defined — they are not.) A hierarchical control scheme must be able to pull out stable perceptual groups as well as to generate new links and tokens so that these stable groups become available for higher-level grouping.

A perceptual group is defined to be any cluster of connected tokens. Specifically, two tokens $i$ and $j$ are in the same cluster if they are either directly connected ($l_{ij} = 1$ or $P_{ij} = 1$) or indirectly connected through other links. When such a cluster is deemed to be stable in terms of the energy minimization process, then we would like to be able to freeze this cluster by exchanging the variable links $l_{ij}$ for permanent ones $P_{ij}$, and then continue with higher level perceptual organization. Pulling out perceptual groups thus entails defining what we mean by 'stable'. Both local and global definitions of stability are being considered.

Once stable perceptual groups are isolated, they should be available for higher level grouping, for example, to discover symmetries or shape similarities on a more global scale. Consistent with the recursive nature of our approach, these stable groups are redefined as primitive tokens, and variable links are generated between them so that they may be further grouped.

## 3.2. RULE BASED GROUPER

The key components of the rule based grouper are:

- A common general format for describing groups
- A classification network for determining the type of group-relationship which exists between objects
- An agenda-based mechanism for determining which groups to apply grouping rules to
- A hierarchical communication scheme for forming groups among potentially spatially separated objects

Groups consist of the following attributes:

- **Bookkeeping Attributes** include the list of inherited group types applicable to a group, the component groups, the groups a group is contained in, and the feature images the group is in register with.
- **Group Specific Attributes** are the properties specific to a given type of group. For a Linear Group this includes attributes such as length and orientation; for a similarity group this includes the average and variance of intensity and contrast.
- **Shape Description** specifies the structural characteristics of a group. Shape descriptions are treated differently than other group specific attributes because they are themselves groups which can be involved in grouping operations. This allows for uniform grouping operations to be applied to groups consisting of disconnected elements.
- **Assimilation Measures** are for evaluating how well all the components of a group fit to the defining characteristics of the group. If the assimilation measures are combined they are constrained to evaluate to between 0 to 1. Note that multiple assimilation measures are associated with a group corresponding to how it was formed with respect to the classification network.
- **Classification Rules** are the conditions used to determine if a group of the specified type should be formed between a pair of groups. This is associated with a node in the classification network corresponding to the group.
- **Extension Rules** specify the conditions under which a group can enlarge itself by adding surrounding groups to it. This includes **Neighborhood Determination Constraints** which determine the areas over which groups can form potential relations with other groups. These fall into a small number of general classes, such as a uniform expanding neighborhood or a directed, cone shape neighborhood.

Figure 5: Group Attributes

The general attributes of perceptual groups are shown in Figure 5. The different types of groups are organized into the attribute inheritance network shown in Figure 6. There is conditional inheritance along the paths in the classification network, so an Interrupted Linear Group inherits the attributes of a linear group, a sequential group, and a similarity group and can selectively inherit the attributes of each of these.

The grouping architecture consists of virtual processors called **grouping nodes** which are organized hierarchically (Figure 7) with respect to the underlying spatially organized data base for extracted perceptual objects. The arrangement of grouping nodes is a basis of communication and applying grouping rules to non-adjacent perceptual structures which are potentially separated by a large distances. Groups at one level can be made available to higher level grouping nodes for the application of rules in a more general context with other groups extracted over a larger area.

The processing flow at each grouping node is organized in a simple loop. Groups are initially obtained by the application of established routines for extracting junctions, contours, and regions. This is followed by a thinning process, the extraction of adjacencies between contours, and fitting linear segments to the resulting contours. This results in similarity, curvature junction, topological junction, linear, and sequential groups. About each group instance a uniform neighborhood is formed to determine pairs of groups. The relationship between each pair of groups is then specialized based upon the classification network in figure 6. Associated with each node in this network is a classification test to see if the groups have the necessary attributes and relations to form a particular type of group. If so, an instance of that type of group is created which inherits the attributes of the parent type of groups. Different types of groups can be created for the same set of component objects.

All the assimilation measures associated with each group instance are then sorted into different lists corresponding to each type of group. There are other also lists for the size of objects and for the number of time steps until a group either indirectly or directly was involved in the application grouping rule. In each of this lists, the best assimilating groups are selected with a bias towards groups that are more specific with respect to the classification network. Three different actions are applied to the selected groups. First, an extension rule which is specific to the type of group is applied which will extend the group. Second, the group forms relations with groups in a uniform neighborhood which are then refined with respect to the classification network. Third, the group forms relations with groups in the entire area of the label plane covered by it's immediate grouping node and is also passed onto a list combining the groups from adjacent grouping nodes.

The newly formed groups are then sorted back into the agendas and processing continues. Processing can continue indefinitely as less and less interesting objects become candidates for the application of grouping rules. Processing can be stopped using criteria such as when there is a sufficiently uniform covering of an image with extracted groups or when all structures belong to unique groups.

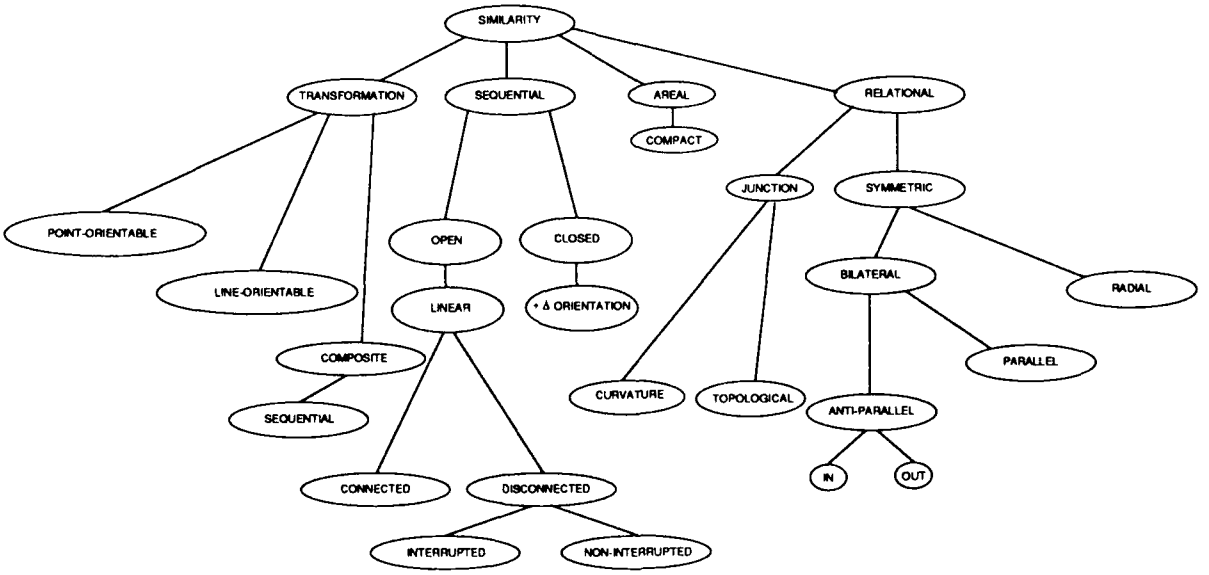



Figure 6: Classification Network

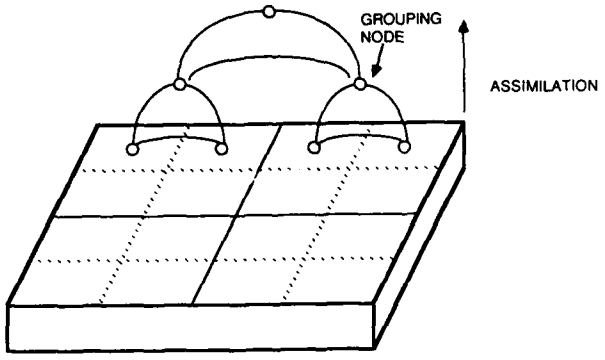



Figure 7: Grouping Node Hierarchy

## 4. IMAGE UNDERSTANDING SOFTWARE ENVIRONMENTS

The software infrastructure we have developed on the knowledge based vision project has been integrated into a uniform and transportable software environment for image understanding, simulation, and terrain modeling [MRL88, MNL87] (see "IU Software Environments" in these proceedings). The environment is organized in terms of four major components: 1) an object-oriented programming mechanism for defining, creating, modifying, and combining objects. This includes common image understanding objects such as curves, images, and histograms. There are mechanisms for defining new objects with automatic inheritance of programming constructs for access, modification, and display. 2) An underlying functional form built from an extendable set of macros that is used for expressing common processing operations for several different types of objects. This functional form corresponds to specifying an abstract parallel architecture in which there is a virtual processor associated with each object. Thus, one can program as though there is a processor at each point in an image. 3) A set of databases which are accessed through a uniform interface for automatically maintaining resources and results during interactive or autonomous processing. There are databases for the storage and access of long term information such as procedures, object definitions, and instances of processing environments. 4) An interactive display facility. We have extended this environment into Common Lisp with a more generic window based interface for portability.

## ACKNOWLEDGMENTS

This document was prepared by Advanced Decision Systems (ADS) of Mounatin View, California, under U.S. Government contract number DACA76-85-C-0005 for the U.S. Army Engineer Topographic Laboratories (ETL), Fort Belvior, Virginia, and the Defense Advanced Research Projects Agency (DARPA), Arlington, Virginia. The authors wish to thank Sachi Toyofuku for providing administration, coordination, and document preparation support.

## REFERENCES

[Bro84] R. A. Brooks. Model-based computer vision. *Computer Science: Artificial Intelligence*, (14), 1984.

[Hop84] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. In *Proc. Natl. Acad. Sci. USA 81*, 1984.

[HR78] A. R. Hanson and E. M. Riseman. Visions: a computer system for interpreting scenes. *Computer Vision Systems*, 1978.

[HT85] J. J. Hopfield and D. W. Tank. 'neural' computations of decisions in optimization problems. *Biol. Cybern. 52*, (141), 1985.

[Kea86] P. K. Kienker and et. al. Separating figure from ground with a parallel network. *Perception 15*, (197), 1986.

[KMY86] C. Koch, J. Marroquin, and A. Yuille. Analog 'neuronal' networks in early vision. In *Proc. Natl. Acad. Sci. USA 83*, 1986.

[LLC*87] T. Levitt, D. Lawton, D. Chelberg, P. Nelson, and J. Dye. Visual memory structure for a mobile robot. In *Proceedings of the AAAI Workshop on Spatial Reasoning and Multisensor Fusion*, Morgan Kaufmann Publishers, Los Altos, California, 1987.

[LLCN87a] T. Levitt, D. Lawton, D. Chelberg, and P. Nelson. Qualitative landmark-based path planning and following. In *AAAI-87 National Conference on Artificial Intelligence*, Seattle, Washington, 1987.

[LLCN87b] T. Levitt, D. Lawton, D. Chelberg, and P. Nelson. Qualitative navigation. In *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, California, 1987.

[LLG*86] D. Lawton, T. Levitt, J. Glicksman, C. McConnell, T. Miltonberger, H. Muller, P. Nelson, and C. Neveu. *Knowledge-Based Vision Techniques Task B: Terrain and Object Modeling Recognition - Final Annual Technical Report.* Technical Report, Mountain View, California, 1986.

[LLM*87a] D. Lawton, T. Levitt, C. McConnell, P. Nelson, M. Black, D. Edelson, K. Koitzsch, J. Dye, T. Binford, D. Chelberg, D. Kriegman, and J. Ponce. *Knowledge-Based Vision Techniques Task B: Terrain and Object Modeling Recognition - Final Annual Technical Report.* Technical Report, Mountain View, California,

1987.

[LLM*87b] D. Lawton, T. Levitt, C. McConnell, P. Nelson, and J. Glicksman. Environmental modeling and recognition for an autonomous land vehicle. In *Proceedings of the DARPA Image Understanding Workshop*, 1987.

[MNL87] C. McConnell, P. Nelson, and D. Lawton. Constructs for cooperative image understanding environments. In *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, California, 1987.

[MRL88] C. McConnell, K. Reilly, and D. Lawton. *Powervision Manual.* Advanced Decision Systems, Mountain View, California, 1988.

# SECTION II

# TECHNICAL REPORTS PRESENTED

# An Integrated Image Understanding Benchmark: Recognition of a 2 1/2 D "Mobile"[1]

**Charles Weems, Edward Riseman, Allen Hanson**

**Computer and Information Science**
**University of Massachusetts, Amherst, MA**

**Azriel Rosenfeld**

**Center for Automation Research**
**University of Maryland, College Park, MD**

## Abstract

This benchmark is the second in a series of DARPA-sponsored efforts to evaluate the merits of various parallel architectures as applied to Image Understanding. The first benchmark exercise considered only execution times for a set of isolated vision-related tasks. This second benchmark exercise addresses the issue of system performance on an integrated set of tasks where the task interactions that are typical of complex vision applications are present. The goal of this exercise is to gain a better understanding of vision architecture requirements, that can be used to guide the development of the next generation of vision architectures.

## 1. Introduction

The need for a computer vision benchmark for parallel architectures has become apparent as researchers from the fields of computer vision and computer architecture have had increasing contact over the last several years. Motion sequences at moderate resolution (512 x 512 pixels) and typical frame rate (30 frames/sec) in color (3 bytes) involves about 23.5 Mbytes of data per second. The amount of computation required for dynamic scene interpretation including the labeling of objects, surface/volume reconstruction and motion analysis is difficult to estimate; however, for many applications computational power in the range of 100 billion instructions per second, plus or minus two orders of magnitude, is probably required. Thus, vision has become a subject of major interest to computer architects.

Unfortunately, the evaluation of progress in vision architectures has been difficult [Duff, 1986]. There are now quite a few interesting machines, both existing and proposed, that may be effective for at least part of the vision problem. However, computer vision transcends a wide range of representations and forms of processing. In addition, despite exciting advances in many of the subtopics of computer vision, there is currently no consensus in the research community on a unified approach to vision. There are many competing approaches and a great deal of debate has persisted. Nonetheless, it is clear that there is a need to address some of the vision/architecture issues in a form that will allow scientific insight and progress in hardware development.

Recent attempts at defining a vision benchmark include the Abingdon Cross problem [Preston, 1986], defined at the 1982 Multicomputer Workshop in Abingdon England, and the Tanque Verde benchmark suite [Uhr, 1986], defined at the 1984 Multicomputer Workshop in Tucson Arizona. The most recent attempt at constructing a benchmark for vision emerged from the DARPA Image Understanding community, where a set of ten vision tasks were defined. These were: Gaussian convolution, zero crossing detection and output of border lists, connected components labelling, Hough transform, convex hull, Voronoi diagram, minimal spanning tree, visibility of vertices in a 3-D model, minimum cost path, and subgraph isomorphism. A meeting was held in November, 1986, in Washington to compare the results of programming, simulating, or estimating the performance of a number of machines on the the individual benchmark tasks. The results were both interesting and thoroughly confusing. The data sets were only loosely specified, leading some to groups report average performance while others reported worst case performance; different groups used different algorithms; some used 32-bit floating point arithmetic while others used 16-bit integer arithmetic, etc. These results must be interpreted with extreme care.

The benchmark defined in this paper is an outgrowth of the first DARPA benchmark, and is again sponsored by DARPA. We have chosen here to address the need for an integrated vision benchmark that transcends several different representations and forms of processing that are typical of complex vision applications. The scientific gain that should result from this exercise is a better understanding of vision architecture requirements, and the performance bottlenecks in different classes of machines, so that the needs of vision processing can be better addressed in the next generation of architectures.

[1]This work was supported in part by the Advanced Research Projects Agency of the Department of Defense under contract number DACA76-86-C0015, monitored by the Engineer Topographic Laboratory.

## 2. Benchmark Philosophy

In writing an integrated image understanding benchmark, the goal is to create an interpretation scenario that is an approximation of an actual image interpretation task. One must remember, however, *the benchmark scenario is not an end in itself, but rather it is a framework for testing machine performance on a variety of common vision operations and algorithms, both individually and in an integrated form that requires communication and control across algorithms and representations.* This benchmark is not intended to be a challenging vision research exercise, and in fact we believe that it *should not* be. Instead, we would like to exercise parallel architectures with a diverse range of operations that are commonly used and in a sequence that requires the sorts of transformations of data and control of processes that occur in a typical interpretation task.

In other words, the benchmark must be interesting and broad, but simple and generic. The problem is to balance these conflicting requirements and yet define a set of tasks that effectively tests the capabilities of different architectures. A second constraint on the development of this benchmark suite is that it must use as many of the algorithms developed for the previous DARPA benchmark of non-integrated vision tasks as possible, in order to take advantage of the programming effort that has already been expended.

We have also attempted to minimize redundant operations with respect to the architectural features that are exercised. The result is that, from an image understanding perspective, the benchmark scenario may be viewed to be somewhat unrealistic and contrived as an interpretation task. However, as a benchmark, it is a valid means of testing performance.

The great variety of architectures that should be tested is itself a complicating factor in designing a benchmark. We recognize that each architecture may have its own most efficient algorithm for computing a given function. However, the tasks must be as well-defined as possible so that the results from different machines will be comparable. We should be testing the performance of the machines rather than the cleverness of their programmers. *Consequently, there should be no short-circuiting of the benchmark tasks* even if the same solution can be achieved by other, possibly simpler, means.

To this end, we are specifying a recommended method for solving each of the tasks. Whenever possible, this is the method that should be used. If the method is inappropriate for a particular architecture, then another method may be used. In such a case, the written summary of results should include a justification for the choice, and an explanation of why the recommended method was not applicable to the architecture. Note that the recommended method may not be optimal. The objective is to use a simple technique so that the results will be consistent and easier to compare. Benchmarkers are welcome to submit timings for more optimal methods in addition to those for the recommended method. However, every attempt should be made to use the recommended method, or as similar a method as possible.

We will be more fully instrumenting each task in the benchmark. In addition to simply measuring raw speed, we will develop a set of quantitative and qualitative measures to be reported for each task, as well as for the entire scenario. Thus, it may be necessary to run the benchmark more than once in order to gather the required results. For example, if it proves difficult for a system to separate the processing time from the time required to output the instrumentation data for an intermediate result, then it may be simpler to rerun the benchmark with the instrumentation output disabled in order to determine processing times. Note that this instrumentation will be specified in a later communication.

In order to make the resulting data more meaningful, we are attempting to constrain the variability in the experiments as much as possible. The benchmark designers (the University of Massachusetts and the University of Maryland) have assumed the responsibility of testing this benchmark using traditional (sequential) methods before parallel programming and analysis begins. This has removed ambiguity and led to more precise statements of the problem, detailed specifications of the benchmark tasks, and code for the sequential benchmark and the generation of image data. The input data sets will be provided to those participating in the study so that all machines will be evaluated on the same test data, and code for generating additional test data will also be distributed. The benchmark designers have the responsibility to produce data sets that test the problem domain fairly, avoiding situations that are unrealistically complex and beyond the reasonable capability of any machine, while testing many interesting empirical situations.

In response to the results of the first DARPA benchmark exercise, we are attempting to develop guidelines for factoring in differences in hardware technology and the scalability of architectures. This will of course, require the specification of technology assumptions by each participating benchmark group.

As a final note, we realize that every benchmark exercise has a tendency to turn into a horse race. However, we hope that the result of this exercise will be the creation of a much more useful set of data and lessons than simply some lap times that are only partially interpretable. The additional constraints and required instrumentation are our attempt to ensure the usefulness of the benchmark results, and provide scientific insight and progress in an exciting as well as confusing research area.

## 3. Features of the Benchmark

- It involves a simple image domain with well-defined, well-behaved objects.
- It requires both bottom-up (data-directed) and top-down (knowledge or model-directed) processing. The top down processing can involve processing of low- and intermediate-level data to extract additional features from the data, or can involve control of low- and intermediate-level processes to reduce the total amount of computation required.
- It tests low-level operations such as convolution, thresholding, connected components labeling, edge tracking, median filter, Hough transform, convex hull, and corner detection.
- It requires utilization of information from two sensors in order to complete the interpretation process.
- It tests grouping operations and graph matching, as representative examples of intermediate-level and high-level processing, respectively.
- It requires the use of both integer and floating point representations.
- It involves the presence of occlusion (ie. absence of data) so that issues of partial matching must be considered.
- It tests the communication channels between the symbolic and numeric levels of processing.

## 4. Overview of the Integrated Benchmark

This benchmark task suite involves recognizing an approximately specified 2 1/2 D "mobile" sculpture composed of rectangles, given images from intensity and range sensors. It is our intention that the test images be designed so that neither, by itself, is sufficient to form a complete match.

The object to be recognized is a collection of rectangles of various sizes, brightnesses, two-dimensional orientations and depths. It can be thought of as a semi-rigid mobile consisting of suspended rectangles floating in space with fixed spatial relationships. To simplify the task, each rectangle is oriented normal to the Z axis (the viewing axis) and the image is constructed under orthographic projection. The model of the object that is provided is approximate in the sense that the sizes, orientations, and depths of the rectangles as well as their spatial relationships are constrained to within some tolerances.

The rectangles that make up the object are interspersed with additional extraneous rectangles in the scene from which the two images are taken. These additional rectangles may occlude portions of the mobile object, and some of the adjacent rectangles in the scene may have very similar brightnesses.

One of the images is from a depth sensor, and the other is from a visible light sensor (B&W). The depth image is a 512 × 512 array of 32-bit floating-point values. The intensity image is a 512 × 512 array of 8-bit integer values. Figure 1 shows an intensity image of a sample object. Figure 2 shows a depth image of the same object. Figure 3 shows the same object as in Figure 1, with extraneous rectangles added. Figure 4 shows a depth image of the same object with extraneous rectangles added. In the depth images (Figures 2 and 4), darker rectangles are closer. (Note that some of the rectangles in the depth images have been lost in the printing process.)

A set of mobile models is provided, any one of which may be present in the images. The goal is to determine which of the models is actually present, the degree to which it is visible (matchable), and to update the model with positional data that has been extracted from the images.

## 5. Description of a Mobile Model

A mobile is a collection of rectangular surfaces that are oriented parallel to the image plane. This results in a constant depth for each rectangle. A mobile can be thought of as a hanging mobile looked at from directly above. The rectangles are suspended perpendicular to the force of gravity by invisible links. The rectangles can have any one of several different intensities.

A mobile model is described by a tree structure that represents the invisible links, increasing in depth, between the rectangles of the mobile. Each node of the tree contains depth, size (lengths of the major and minor axes), orientation, and intensity information for a single rectangle. The links of the tree describe spatial relationships between certain pairs of rectangles. Each model link represents the two-dimensional projection of a three-dimensional mobile link into the plane of the parent rectangle and is described by a bearing and a distance from one rectangle's center point to another rectangle's center point. Thus the spatial relationships between rectangles are specified in the $X$ and $Y$ dimensions by the links, and in the $Z$ dimension by the depths of the nodes.

Each rectangle defines a new coordinate system for its child links that is relative to its own position. The center of the parent rectangle becomes the origin of the coordinate system for links emanating from it. The model's coordinate system is rotated and shifted with respect to the image coordinate system.

The model given does not exactly represent the mobile pictured. The links in the mobile can stretch and sway. This means that the length and bearing of a mobile link

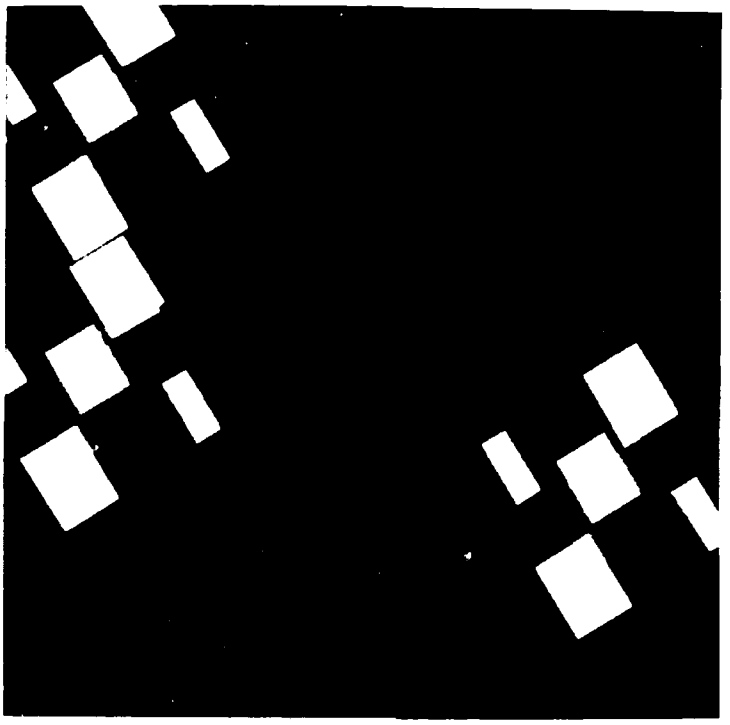


Figure 1. Intensity Image of Example Model

Figure 2. Depth Image of Example Model

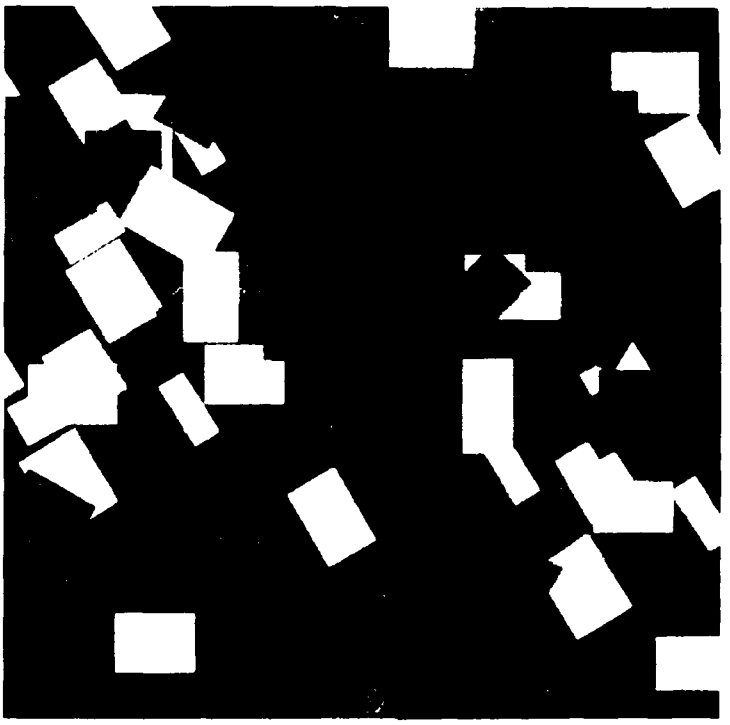


Figure 3. Example Benchmark Intensity Image

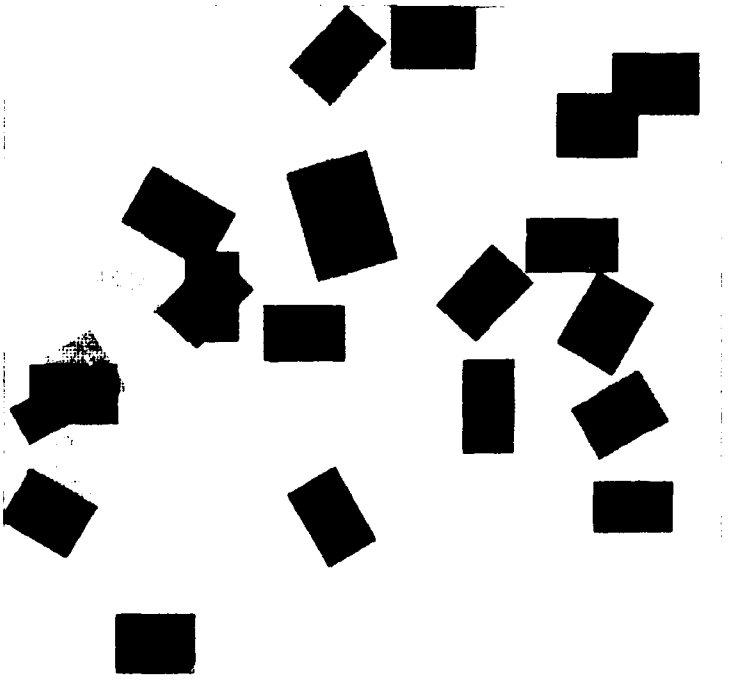


Figure 4. Example Benchmark Depth Image
(before addition of sensor noise)

may be slightly different from the nominal values given in the model. Further, the sizes and depths of the recentangles are given with an associated imprecision. Intensity is not subject to any variation from the model-specified nominal values.

An instantiation of the model is created by randomly perturbing some of the parameters of the model: a) the nominal depth, size, and orientation of each rectangle, and b) the nominal angle and distance of each link given in the model. The perturbations are obtained by sampling a random Gaussian distribution (truncated at $\sigma = 3$). The scene is then created by orthographically projecting an instantiation of the model onto the image plane.

Note that since each link is perturbed individually, the position of a rectangle connected by a single link to a known rectangle is constrained to a small area. However, rectangles that are two or more links from a known rectangle have significantly greater uncertainty in their relative position, and thus are constrained only to a much larger area.

A set of error factors (tolerances) that define the maximum perturbation of each model element's parameters will be provided with each data set. The different error factors are:

$E_{xy}$ = Error in $X$ and $Y$ position (link angle and distance)
$E_L$ = Error in rectangle size (axis lengths)
$E_D$ = Error in depth

Note that $E_{xy}$ specifies the error factor for both a link's angle and its distance. This is because $E_{xy}$ is actually the distance that the child rectangle can shift in the $X - Y$ plane, relative to the nominal position specified by the link. $E_{xy}$ may thus be thought of as the radius of an error circle, centered at the nominal position of the child rectangle, which defines the maximum $X - Y$ displacement of the center of the instantiated child rectangle.

Roughly ten mobile models will be provided with the test images and it is the goal of the benchmark processing to determine which model best matches the image data, and to update the positional data in that model.

## 6. Description of Intensity Image

The intensity image consists of an array of 512 × 512 8-bit pixels. The upper left pixel in the image is assumed to have the coordinates (0,0) in the discussion that follows, with the X coordinate increasing to the right and the Y coordinate increasing downward. The rectangles in the image appear against a black (level 0) background. Each rectangle will have a constant gray level, selected from a small number of levels (say 8). The intensity image is noiseless and created as if only one ray from the scene reaches every pixel: the ray perpendicular to the image plane and through the center of that pixel. Therefore, there is no aliasing and all rectangles form sharp boundaries with any occluding or occluded rectangles of different intensity, and with the background.

In addition to the rectangles that are present in the model, the scene contains spurious rectangles that occlude, or are partially occluded by, portions of the model. All sorts of coincidental alignments of rectangles may occur. In general, it will be impossible to completely extract, via data-directed processing, all of the object model rectangles from the intensity image alone since some rectangles may be occluded (partially or completely) by others. It may also be the case that two rectangles of the same gray level are adjacent or overlap. Thus, it is possible that from 1 to 4 of the corners of any given rectangle may not be visible, thereby producing ambiguity in the matching process. Note that it will be possible to extract some of these rectangles from the depth data even though they are hidden in the intensity data. For example, a large spurious rectangle might be located behind a smaller model rectangle with the same gray level, so that they are distinguishable in the depth image, though not in the intensity image.

## 7. Description of Depth Image

The depth image is a 512 × 512 array of 32-bit floating point values (IEEE standard representation). A larger pixel value indicates greater depth. The coordinate system is the same as that of the intensity image, and the depth image is registered with the intensity image. The background is at a constant depth which is greater than the depth of any rectangle in the image. At first, a noiseless image is created by drawing the spurious and model rectangles in greatest to least depth order. Because the invisible links of a mobile model are similar in length, the result is a clustered distribution at several principal depths. Within each depth cluster, the depths of the individual rectangles differ by just a small amount relative to the differences in depth between clusters. The spurious rectangles are also present in the depth image and have roughly the same distribution as the model rectangles. Simulated Gaussian sensor noise is added to the noiseless depth image to produce the actual test data. The distribution of the noise is scaled and truncated so that individual pixels will have a depth error that at most causes them to appear at different depths within a cluster but is not so great that they will appear to belong to rectangles of a different depth cluster. Thus, it is possible for a pixel in one rectangle to have a depth value that appears to make it a member of an adjacent rectangle in the same depth cluster, and therefore depth boundaries will be unreliably extracted.

The placements and depths of the spurious rectangles will be such that some of the model rectangles may be impossible to extract from the depth image. In the intensity image, however, these rectangles may be easier to extract.

For example, a large spurious rectangle may be placed only slightly behind a smaller model rectangle, so that the difference in their depths is lost in the noise, but the two rectangles might differ significantly in gray level.

# 8. General Overview of the Processing Scenario

All of the processing steps given must be included in the parallel versions of the benchmark. Furthermore, the general scheme of the processing described at each step should be followed. Included in the description of the steps are thresholds and other parameters, which should be used as given.

Processing begins with some low-level operations on the intensity and depth images, followed by intermediate-level grouping. Given these initial results, the remainder of the benchmark task involves the following steps: Initial rectangle hypothesis generation, initial model matching, top-down depth verification for the initial match and extension of the model match for disambiguation and model update, and result presentation.

## 8.1 Initial Processing of the Intensity Image

The low-level operations on the intensity image consist of identifying connected components and finding the corners of each connected component based on a K-curvature operation. The initial processing of the intensity image also includes an intermediate level grouping operation that consists of creating good hypotheses for rectangles from the lists of corners around connected components in the image. The result of the initial intensity image processing is thus a set of connected component tokens. The only feature that is extracted from each connected component as a whole is its intensity. However, each component region has associated with it a list of the corners that were extracted from its boundary.

## 8.2 Initial Processing of the Depth Image

The low-level operations on the depth image consist of smoothing via median filtering, computing the magnitude of the gradient, and thresholding the gradient magnitude. The result of the initial depth image processing is an image array that represents points in the depth data that have large gradient magnitudes. The smoothed depth image is also used in later stages of processing.

## 8.3 Intermediate-Level, High-Level, and Top-Down Processing

Intermediate level processing starts with bottom-up grouping of right-angle corners, in component tokens, in order to generate rectangle hypotheses. The resulting candidate rectangles form the basis of the initial model graph matching operation. The intermediate level operation on the depth image is a top-down directed search for expected rectangles, incorporating a spatially local Hough transform with model-constrained ranges on the parameters for each rectangle. The high level operations are first, constrained subgraph-to-subgraph matching to choose and orient the models to be matched; and second, top-down control of probes into the depth and intensity images to find and fix the parameters of the rectangles that are required to fill out the chosen models. As a concluding step, an image is produced that represents the single best model match as an overlay with the original intensity image.

The goal of the first graph match step is to attempt to establish the most likely positions and orientations of the modeled objects in the image and possibly to eliminate some of the models from further consideration for matching. Since only some of the model rectangles will have been extracted from the intensity image, portions of each model will have no match. The unmatched portions of a graph model are used to focus attention in the depth image so that additional localized features can be extracted and the model can be extended through the use of context. This match extension step is further divided into three parts that are repeated for each model rectangle: model directed rectangle detection, rectangle depth and intensity verification, and model update.

# 9. Low-Level, Bottom-Up Processing

## 9.1 Intensity Image

### 9.1.1 Label Connected Components

**Task:** Each connected component in the image is given a unique integer label. A connected component is any contiguous collection of pixels that have the same intensity level value. Contiguous is defined as adjacency in any of four directions (N,S,E,W). The result is an image-size array where each pixel in a component has the value of the component label.

**Recommended Method:** Each pixel has an address associated with it that is the concatenation of its binary row and column number (in a 512 × 512 image, the

resulting address is an integer with 18 significant binary digits, the column number being the low order 9 bits of the address). Each connected component takes as its label the lowest pixel address value from among the members of the component. This may be done by propagating the lowest address from the pixel at that location throughout the component. The order in which the propagation proceeds (ie. radially, by rows or columns, through distance doubling, etc.) is left to the choice of the benchmark programmer.

### 9.1.2 Compute K-curvature and Extract Corners

**Task:** For each connected component, compute the K-curvature of the boundary of the component. (The boundary is the set of pixels that are members of the component and are adjacent (N,S,E,W) to at least one pixel that is a member of another component). The K-distance is a parameter supplied with the data set (in the test set, a distance of 4 is used). The K-curvature values are then smoothed using a one-dimensional Gaussian mask that is 7 elements wide in order to eliminate multiple local peaks near a corner. The first derivative of the smoothed curvature values is computed and zero-crossing points along the derivative of the boundary are determined. Corners are defined to be those points in the smoothed curvature values that exceed a specified threshold (ie, have high curvature) and that correspond to a zero-crossing (in order to select the point with maximum local curvature).

**Recommended method:** K curvature is computed for each border pixel by tracking along the border pixels of each region for a distance of K (equal to 4 in this case) in each direction. A scaled value is computed that is proportional to the angle formed by the straight lines linking the center pixel with its neighbors that are K edge points away in each direction. Relative angles between these lines are computed with a table lookup operation, using the differences of the X-Y pixel coordinates of their end points to index into a two-dimensional array of scaled angles. The angles for the two lines are then combined to determine the inside angle between them. For K=4, the scaled angle array used in the sequential implementation is (the scaling factor is Degrees/9, with angles oriented to correspond to the image coordinate system):

The combining function is:

$\theta = | \theta_1 - \theta_2 |$

If $\theta > 20$

Then $\theta = 40 - \theta$

Any point with $\theta = 10 \pm 2$ is flagged as a right angle (this flag will be used in the rectangle hypothesis generation step). An unnormalized one-dimensional Gaussian smoothing function is computed by forming the sum of the products of the angle values of the edge pixels within each 7-wide window (3 pixels in each direction from the center of the window) along the border. The window coefficients are:

3, 27, 90, 136, 90, 27, 3

The first derivative of curvature is computed by convolving the border with a mask of -1, 1 (i.e. computing the difference between each pixel on the border and one of its neighbors). Zero crossing points are computed by comparing the signs of neighboring border pixels. Corners are located by thresholding the smoothed curvature values and forming the logical AND of the resulting binary image with the binary image that results from the zero-crossing detection. (A curvature threshold of 1000 was used with the supplied test data.)

## 9.2 Depth Image

### 9.2.1 Smoothing

**Task:** Smooth the depth image while preserving edges by applying a median filter with a $3 \times 3$ mask to the original depth data.

**Recommended method:** A new image is created where each pixel is the median value of the pixels in a $3 \times 3$ window centered on the same location in the original image. Pixels that are adjacent to the image boundary should simply retain the same values as in the original image.

### 9.2.2 Gradient Magnitude Computation

**Task:** Compute a standard $3 \times 3$ Sobel operation on the smoothed depth image. Pixels beyond the edge of the image should be treated as having values of zero. The gradient magnitude is the square root of the sum of the squares of the X and Y magnitudes that result from the Sobel.

**Recommended Method:** Any standard method is acceptable.

### 9.2.3 Threshold

**Task:** Select strong edge pixels by thresholding the output of the Sobel operation at a specified level. The result is a binary image where a 1 represents an edge pixel.

**Recommended Method:** Any standard method is acceptable. A threshold of 500 was used for the test data.

# 10. Intermediate-Level, High-Level, and Top-Down Processing

## 10.1 Rectangle Hypothesis Generation

**Introduction:** The goal of the initial rectangle hypothesis generation step is to find good candidate rectangles among the components of the intensity image so that model matching can begin. Note that the goal does not require every possible candidate rectangle to be found. Only some rectangles that can be reliably used in the matching process are of interest, because once an initial match is formed the remainder will be extracted by top-down processing of the depth data. This can be viewed as using strong cues to focus attention in the subsequent extraction of weaker or more ambiguous image events.

The result of rectangle hypothesis generation is a set of candidate rectangles. Each rectangle is described by six parameters: the coordinates of its center pixel, the lengths of its major and minor axes, the orientation of its major axis, and its intensity.

**Task:** Extract good quality rectangles with three or four corners visible in the intensity image. The input (for each connected component) is a list of the row and column coordinates of center points for corners that were detected on a component's boundary. Additionally, corners whose K-curvature was calculated to be close to 90 degrees are flagged. Corner points that are not on the convex hull of this point set are discarded. For each corner point on the hull, the angle formed by the computed lines that connect it to its neighboring corner points is computed. A component is declared to be a rectangle candidate if there are at least three contiguous right angle ($\pm$ two degrees) corners on the convex hull, which were also measured as right angles in terms of K-curvature. Two of the opposing corners are used to compute the center of the rectangle (which is taken to be the pixel nearest the midpoint of the diagonal line segment defined by the two corner points). The length of the major axis is the greatest distance between adjacent pairs of the rectangle's right angle corners. The orientation of the rectangle's major axis, relative to the origin, is also computed. The length of the minor axis is the minimum distance between adjacent right-angle corners.

**Recommended Method:** If the K-curvature has not flagged at least 3 corners as right angles for a component, then reject the component as a rectangle. Otherwise the convex hull should be computed with a parallel Graham Scan type of algorithm. If there are less than 3 flagged corners on the hull, then the component is rejected as a rectangle. Next, corners of the hull that were flagged as right angles have their hull angles checked — i.e., angles formed between neighboring vertices of the hull are checked to determine whether or not they are approximately right angles. This is easily accomplished by exploiting the relationship between the vector dot product and the cosine of the angle as follows:

Given a triangle of points a,b,c with $\theta$, the angle in question at b, then each point defines a corresponding vector A,B,C, from the origin to that point. Then we have

$$\cos\theta = \frac{(A-B)\cdot(C-B)}{|A-B|\,|C-B|}$$

squaring, we get

$$\cos^2\theta = \frac{((A-B)\cdot(C-B))^2}{((A-B)\cdot(A-B))((C-B)\cdot(C-B))}$$

Because there is an angular tolerance ($E_\theta$) the final predicate becomes

if $\cos^2\theta <= \cos^2(90 + E_\theta)$
AND $\cos^2\theta >= \cos^2(90 - E_\theta)$
then right_angle
else not right angle

This method avoids the use of inverse trigonometric and square-root operations. Note that the squares of the cosines for the range test can be assigned to global constants, since they are not affected by the particular set of points. If there are three successive right angles then the component is labeled a rectangle, otherwise it is rejected. Finally, if the component is a rectangle, its parameters are determined from the triangle defined by the three successive right angle corners as follows:

major-axis = length of the longer side
minor-axis = length of the shorter side
orientation = angle of the major axis with respect to the X-axis
center = pixel coordinates nearest the midpoint of the hypotenuse

## 10.2 Graph Matching

**Introduction:** The candidate rectangles can be thought of as forming a complete graph, where the links of the graph represent hypothetical mobile links between every pair of rectangles. Because the model is described by a graph structure, the matching process takes the form of finding the maximal set of constrained subgraph to subgraph isomorphisms between the complete graph formed by the candidates and the model graph. The matching process thus tries to match candidate rectangles to model rectangles by their size, intensity, and depth; and to match hypothetical links to model links by comparing the spatial relationships between rectangles that the links represent. Thus, to

establish a subgraph isomorphism requires that a connected set of candidate rectangles and links be found that has the same properties and spatial relationships as some connected set of model rectangles and links.

Basically, the strategy that is followed is to find all of the single node isomorphisms (ie. matches between rectangles alone) and join them in pairs to form isomorphisms with a radius of one link, and then to merge those isomorphisms into larger subgraph isomorphisms. Note that initially there may be multiple matches to each node; however, as spatial relationships are used to further constrain the matches, these will be quickly reduced to the point that only a small number of possible matches remain. The test data set has been purposely designed so that the remaining ambiguous matches can only be fully resolved by analysis of the depth image. Once an initial set of matches has been found, the depth data is probed top-down to verify that the matched rectangles are at the proper depth as specified by the model. This step may eliminate some of the ambiguous matches, but not all.

After the graph-matcher confirms the depths of the rectangles that were found in the intensity data, it continues trying to merge matched fragments to form larger subgraph isomorphisms. Whenever it is necessary to extract a rectangle from the data in order to extend a match, the graph-matcher probes the depth data in a top-down manner in order to locate a model rectangle that has not been found in the intensity image. The graph matcher also confirms the rectangle's depth and match strength using the routine described in section 10.3. Each rectangle that is established to exist in the depth data results in an update to the rectangle's position parameters in the model.

**Task:** Given the set of mobile models and the initial rectangle hypotheses from the intensity processing, form an initial subgraph to subgraph match of the rectangles and models. The initial match process starts by matching candidate and model rectangles (by size and intensity) and then tries to extend each match to a radius of one link (by comparing the spatial relationships of pairs of matched candidate rectangles to those of the corresponding pairs of adjacent model rectangles). Then confirm the model-specified depths of and intensities the matched rectangles by directed probing of the intensity and smoothed depth data (see section 10.3). Extend the matches for each model by model-directed, local probing of the thresholded gradient magnitude image for the presence of edges that correspond to each model rectangle (see section 10.4). The model-specified depth and intensity of each rectangle that is found by the probe is then verified by probing the intensity and smoothed depth images using the same technique as for confirming the depths of the initial rectangles (see sections 10.3 and 10.4.3).

The top-down probing of the depth data results in confirmation or rejection (veto) of the presence of each model rectangle in the depth data. Each confirmed rectangle is used to extend the model match, and update the positional information in the model. A rejected rectangle results in the rejection of whatever portion of a match depends on it. It is possible for an entire model to be rejected because one of its rectangles is rejected. For example, if previous processing has fixed the position of the model in the image so that there is only one possible pose for the rectangle, and it is rejected by the depth probe, then there is no alternative but to reject the model. (Note that a rectangle is only rejected if there is strong evidence that it is not in the image. Occlusion does not cause a rejection).

Thus, in the process of extending a subgraph match, if it is necessary to link to a rectangle that is rejected, the entire subgraph will be labeled as rejected. Once a subgraph is rejected, no further attempt is made to extend it. However, it is possible that an attempt will be made to extend another subgraph to link with the rejected subgraph. If the other subgraph requires that link in order to be extended, then it too must be labeled as rejected. In this way, the rejection of a single rectangle can spread to whatever portion of a proposed match depends upon that rectangle.

Once verification has been completed for each model rectangle, and a global match has been computed for each model that is not rejected, an overall model match strength (the average of the node match strengths) is computed for each model. The model with the highest match strength is declared the best match and is used to produce the final output image.

**Recommended Method:** We begin with some definitions. In addition to the model graphs there are two globally maintained lists, called the probe list and the active root list. There are also two lists associated with each model rectangle, called the pose list and the compatible rectangle list.

Probe list: Contains a list of all model rectangle poses which may have empty link fields that await matching. The list is created as a result of the initial matching process and is used to guide the match extension process. The list is maintained in order according to the match strength of each rectangle pose. Thus, strong cues are given priority in the matching process. When a rectangle at the front of the probe list has all of its links filled in, it is removed from the probe list.

Active root list: Contains a list of all model rectangle poses that correspond to roots of models and that

have not been eliminated through a veto. In other words, this is a list of all rectangles that are potential roots.

Compatible rectangle list: Each model rectangle has a list of the initially extracted rectangles that match its size and intensity. These lists are created from the set of initial candidate rectangles at the start of the initial graph-match process. Roughly square rectangles (those that still match the model rectangle after being rotated 90 degrees) are added to the list twice (once with the 90 degree rotation). Each rectangle that is added to the list is also duplicated with a 180 degree rotation to cover the symmetric orientation case.

Pose list: For each model rectangle a list of its possible poses in the scene is formed from the compatible rectangle list. The initial pose lists are created during the initial match phase and contain those compatible rectangles with at least one link to a compatible rectangle in a neighboring model rectangle's list of compatible rectangles. Thus, if two neighboring rectangles are linked, they will each appear in the pose list for their corresponding model rectangle.

The poses are the elements that will be linked together by the graph matcher as it tries to build isomorphic subgraphs. Poses either join an isomorphism to extend it, or are vetoed as a result of top-down probing of the depth and intensity data.

Given these definitions, the following five steps describe the initial matching process:

1. Build the compatible rectangle list for every model rectangle: Add to the compatible rectangle list of each model rectangle a copy of the parameters for each candidate rectangle whose size (within $E_L$) and intensity match the model rectangle. If a rectangle also matches the model after being rotated 90 degrees, then add a second copy to the list with that rotation. Lastly, add a second copy of each rectangle with a 180 degree rotation.

2. For each rectangle in the compatible rectangle list of each model rectangle, attempt to establish a link to every compatible rectangle associated with each neighboring model rectangle. A link is formed between two compatible rectangles if their spatial relationships (relative orientation, direction and distance) match the spatial relationship specified by the link between the corresponding model rectangles. Each successful linkage is added to the pose lists of the model rectangles. If one compatible rectangle is linked to another that is already on a pose list, then the new link should be merged into the existing pose rather than becoming a new pose. By merging these poses, larger subgraph isomorphisms are created during the initial matching process, thus saving time in later stages. (Note that, initially, any symmetric pair of rectangles will be linked as two possible poses by virtue of the fact that the two rectangles are in the compatible rectangle lists of both model rectangles. A symmetric pair of rectangles would be two rectangles that have the same size and intensity and are oriented so that a 180 degree rotation of the pair is indistinguishable from the non-rotated pair.) The result of this step is the creation of pose lists, where each pose is an isomorphic subgraph that is created from the compatible rectangle lists. Many compatible rectangles will fail to link to any neighboring rectangles, and will be omitted from the pose lists. For this reason, the compatible rectangle lists will again be checked as part of the match extension process.

3. Form the initial, unordered probe list and active root list all of the rectangles in the pose lists.

4. For each rectangle on the probe list, probe the depth and intensity data for veto or match strength. If the rectangle is rejected its entire linked subgraph is removed from the probe and active root lists, and marked as rejected in the pose list and compatible rectangle list of the appropriate model rectangle. If the rectangle is not vetoed, a match strength is associated with it and it remains on the lists. When a rectangle has been probed, the corresponding rectangle in the compatible rectangle list is flagged as having been probed and is assigned the associated match strength.

5. The probe list is sorted by match strength so that the first element of the list has the greatest strength.

This concludes the initial match phase. Models that have completely empty pose lists will be rejected at this point. Note that the probe list may contain rectangles with very low match strength, because such rectangles are not necessarily vetoed. For example, a rectangle may be mostly occluded, in which case it will have a low match strength, but there is insufficient evidence to conclude that it is not present. Such a rectangle may be thought of as being hallucinated by the system. Because hallucinated rectangles are last on the probe list, they are only used to fill in missing links near the end of the match extension process.

The match extension process consists of the following four steps that are repeated until the probe list is empty.

6. The probe list is searched for the first empty link in a rectangle. Any rectangles at the front of the list that have all of their links filled are removed from the probe list. An empty link corresponds to a link in the model, and thus to a neighboring rectangle in

the model. Note that all possible links between the initial set of rectangles have been found at this point. Thus, an empty link indicates a rectangle that must be added via the match extension process.

7. The object of this step is to make sure that there is a probed data rectangle corresponding to the selected neighboring rectangle. First, the neighbor's compatible rectangle list is searched. If a compatible rectangle is found that has been previously probed, then it should be used. If a compatible rectangle is found that hasn't been probed yet, then it must be probed. If no compatible rectangle is found, use the model parameters for the neighbor to direct a top-down search (local Hough transform) and then probe the area indicated by the search. Add the rectangle to the compatible rectangle list for the neighboring model rectangle, even if the search and/or the probe fails.

8. Once a probed data rectangle has been found for the neighbor, check the neighbor's pose list for any links to it. If a link is found, then the rectangle is part of a larger subgraph isomorphism which should be linked to the isomorphism that contains the pose at the top of the probe list. Otherwise, create a new pose consisting of just the newly probed rectangle and link the pose at the top of the probe list to it.

9. If the neighbor rectangle, or its instance on the pose list is flagged as vetoed, delete the entire linked subgraph from the probe list (and the active root list, if necessary). Also, label as vetoed the entire structure associated with the model pose, corresponding to the probe rectangle. If it was not vetoed, and it was created and/or probed in step 7, then insert the new rectangle in the probe list according to its match strength and also add it to the active root if it corresponds to a model root.

Once the probe list is empty, the match extension process is complete. The result is a set of one or more model matches that are pointed to by the active root list.

## 10.3 Top-Down Probe for Confirmation or Veto of Initial Match

**Introduction:** For each rectangle that is found in the intensity data and matched in a model, the smoothed depth image and the original intensity image are probed to determine a match strength. The probe initially examines the area of the smoothed depth image that corresponds to the rectangle in its hypothesized pose. Pixels are counted that are found to be too distant with respect to the model-specified acceptable depth range. If the count exceeds a threshold then the match is vetoed. If the match is not vetoed, then an area of the intensity image is also probed that corresponds to the rectangle in its hypothesized pose. The result of the depth and intensity probes is a match strength indicated by the percentage of pixels in the area that are within the acceptable depth range and that have the correct intensity, minus the percentage of the depth pixels that are too deep.

**Task:** Given the parameters for a rectangle, probe the smoothed depth data and the intensity image to confirm that a region of roughly the correct depth and intensity is present. There are two parts to this task: a check for a strong indication that the rectangle is not present in the depth data, and the computation of a match strength.

For the first part of this task, a window of the smoothed depth image is selected that corresponds to the hypothesized pose of the rectangle. Within this area, the pixels are divided into three categories: those pixels that are deeper than the allowable range of depths for the rectangle, those that are within the allowable range, and those that are too close. The number of too-deep pixels is determined, and if it exceeds a threshold, the rectangle is vetoed because it is either missing or not in the correct position.

The threshold is equal to

$$(E * E_{xy} + E_L) * (L + l)$$

where

| | | |
|---|---|---|
| $L$ | = | Length of major axis |
| $l$ | = | Length of minor axis |
| $E_{xy}$ | = | Error in $X$ and $Y$ position |
| $E_L$ | = | Error in axis lengths |
| $E$ | = | Error multiplier for this match |

(The value of $E$ reflects the effects of previous processing on the potential error expressed by $E_{xy}$. In this case, because the position was actually computed from the intensity data, the positional accuracy is quite good and $E_{xy}$ can be reduced. Thus $E$ is equal to 0.25 for this match.) In other words, the threshold is the maximum portion of the probe window in which too-deep pixels could appear because of the position and size errors that are created when the model is instantiated. Note that the above formula actually computes a slightly higher threshold than necessary in order to account for orientation error without performing additional computations involving angles.

If the rectangle is not vetoed, then a second step computes a match strength for the entire hypothesized rectangle pose area in the smoothed depth image and

the intensity image. The match strength is reported as the percent of pixels that are both in the correct depth range and have the correct intensity minus the percentage of pixels that are too deep.

**Recommended Method:** In the smoothed depth image, within a window that corresponds to the hypothesized pose of the rectangle, the pixels are thresholded at $D + E_D * D$. Pixels that exceed the threshold are too deep, and are counted. If their number exceeds $(E * E_{xy} + E_L) * (L + l)$, then the rectangle is vetoed.

If the rectangle is not vetoed, then a range test is performed on the smoothed depth image within the same window. The acceptable range is $D \pm E_D * D$, and pixels that pass the test form a binary image. Within a registered window of the intensity image, pixels are tested for intensity equal to $I$ (the model rectangle's intensity). Another binary image results, which can then be combined via a logical AND with the other binary image to produce a third image representing those pixels that have the correct intensity and depth. The match strength for the rectangle is then computed by dividing the number of these correct pixels minus the number of too-deep pixels by the total number of pixels in the window area.

Note that if a rectangle is fully occluded, it will not be vetoed but it will have a match strength of zero or less.

## 10.4 Top-Down Examination of Depth Data

**Introduction** When it is necessary to search for a model rectangle in the depth data, a Hough transform is applied top-down within a spatially local window of the thresholded gradient magnitude image, followed by a search of the Hough array to locate evidence for parallel and perpendicular lines that form a rectangle with the appropriate size and orientation.

The rectangle location strategy is as follows: an X-Y oriented window that is large enough to contain the rectangle, given its possible position and size errors, is computed and used to mask the thresholded gradient magnitude image that resulted from the initial depth processing. Within that window, a Hough transform is computed in order to detect lines. The Hough array is then searched for the maximum bucket with approximately the same orientation as the major axis of the model rectangle. If a sufficiently strong bucket is found, the corresponding line is used to anchor the search for the other three edges of the rectangle. At least two of the remaining three edges must be detected as lines in the Hough array in order for a positive search result to be returned. If only three of the edges are detected, the fourth is inferred from the model data.

### 10.4.1 Depth Window Size and Position

The rectangle search is given the position and size of an X-Y oriented search window and the expected orientation for the rectangle's major axis. The window's bounding edges are computed from the following formulas:

| | | |
|---|---|---|
| $X$ | = | $X$ position of rectangle's center |
| $Y$ | = | $Y$ position of rectangle's center |
| $L$ | = | Length of major axis |
| $l$ | = | Length of minor axis |
| $\theta$ | = | Orientation of major axis |
| $E_{xy}$ | = | Error in $X$ and $Y$ position |
| $E_L$ | = | Error in axis lengths |
| $E_\theta$ | = | Error in orientation |
| $E$ | = | Error multiplier for this match |
| $S$ | = | $sin(\theta)/2$ |
| $C$ | = | $\lvert cos(\theta)/2 \rvert$ |

When the Hough transform is being used to locate a rectangle, the positional information that is used to place the window is based on the corresponding model rectangle. However, the estimated position of the model rectangle is relative to another rectangle (the rectangle at the top of the probe list) whose position has already been fixed in the image. Thus, the positional accuracy for the search depends on the accuracy of the position of the rectangle at the top of the probe list. If the match strength of that rectangle is high ($\geq 0.5$) then the positional accuracy is probably good and $E$ is equal to one. If its match strength is low, then its positional accuracy is probably poor and so $E$ is increased to two.

Window starting row=

$Y - E*E_{xy} - S*(L + E_L) - C*(l + E_L)$
(or 0 if result is $< 0$)

Window ending row=

$Y + E*E_{xy} + S*(L + E_L) + C*(l + E_L)$
(or 511 if result is $> 511$)

Window starting column=

$X - E*E_{xy} - C*(L + E_L) - S*(l + E_L)$
(or 0 if result is $< 0$)

Window ending column=

$X + E*E_{xy} + C*(L + E_L) + S*(l + E_L)$
(or 511 if result is $> 511$)

The rectangle locating routine searches within this window for a rectangle with its major axis oriented in the direction specified ($\theta \pm E_\theta$) The search consists of the steps in the following section.

### 10.4.2 Hough Transform and Rectangle Search

The Hough transform was chosen for this step for several reasons. The Hough transform tests some very interesting aspects of parallel architectures and is thus useful from a benchmarking point of view. Because it was previously implemented for the original benchmark, it should take less time to adapt it than would be required for implementing an entirely new operation. Also, several of the reviewers of the draft benchmark specifically requested that it be included. The constrained, verification mode of application of the Hough transform was arrived at after considerable experimentation. For example, initial attempts to apply the Hough transform to the entire image met with little success because the numerous strong straight edges combined to produce many false peaks and masked most of the true edges. Constraining the transform to a small window significantly improved the results, and further emphasized the top-down control aspect of this step. Also, the use of gradient orientation information from the Sobel was tried, but it was found that the results were not sufficiently improved to warrant the additional complexity.

**Task:** Perform a model-directed search of the thresholded gradient magnitude image, in order to locate edges that correspond to a hypothesized pose of a model rectangle. The search begins with a Hough transform that is applied to the image within an X-Y oriented window that is sized and positioned so that it will contain the entire rectangle, even if it is subject to maximum errors in position, orientation and size.

The Hough array is then searched for strong lines that form a rectangle in the appropriate size and orientation ranges. If only one line along an axis is found, the missing parallel line is inferred from the model data. New position, size, and orientation values for the rectangle are computed from the extracted lines. If both lines parallel to a given rectangle axis are missing, the original model data is returned unchanged.

**Recommended Method:**

1. Compute the Hough transform of the binary image (from step 9.2.3) within the specified window. The range of the Hough space is $0 \leq \theta < 360$, or $-\sqrt{2} * 256 \leq rho \leq \sqrt{2} * 256$, however the accumulator array has a resolution of $H_\theta$ per bucket in $\theta$, and $H_\rho$ pixel widths per bucket in $rho$ (Where $H_\theta$ and $H_\rho$ are parameters supplied with the test data. Also note that, if desired, the window dimensions may be used to further constrain the range of $rho$, and the specified orientation range can be used to constrain the ranges of $\theta$. Thus, it is possible for the size of the Hough array to be greatly reduced if that is desirable.

2. Search the Hough array to determine the maximum accumulator with $\theta$ in the allowable ranges for the rectangle's major axis orientation ($\theta \pm E_\theta/2$) and ($\theta + 180 \pm E_\theta) MOD 360$). The minimum allowable count in the selected accumulator is $H_{t1}$ If there is no accumulator with a count of at least $H_t$, then the rectangle routine will report that no rectangle could be found. (The values of $H_t$ and $H_{t1}$ are supplied as parameters with the test data.)

3. If a major-axis line is found, the search proceeds by looking for a parallel line. Such a line must be at least $l - E_L$ distant from the first line, have a count of at least $H_{t2}$, and $\theta$ equal to that of the first edge. ($H_{t2}$ is another parameter that is supplied with the test data.)

4. The next step is to look for the strongest perpendicular line in the Hough array. The selected accumulator must have a count of at least $H_{t1}$ and be within $E_\theta/2$ of perpendicular to the expected major axis orientation of the rectangle.

5. The last line must be found at least $L - E_L$ distant from the third line, have a count strength of at least $H_{t2}$, and have the same $\theta$ as the third line.

6. If, for either axis, no corresponding line is found, the rectangle routine reports that no rectangle is present in the window.

7. If 4 lines were found, the routine computes the new $X$ and $Y$ position of the center, the lengths of the major and minor axes, and returns these values together with the orientation of the major axis.

8. If only one line is found parallel to an axis, the length of the perpendicular axis is not updated.

### 10.4.3 Top-Down Probe for Confirmation or Veto

This is the same process that is used in section 10.3.1 for the initial depth confirmation. However, the rectangles that are being confirmed are those that were detected in the depth data, rather than the intensity data. Because the rectangle parameters are identical in form, regardless of the source of the rectangle, the same task description and method as in section 10.3.1 apply here.

## 10.5 Result Presentation

**Task:** Create a display that indicates those model rectangles that were strongly matched and those that were not. The display will consist of the original intensity image with model rectangles brightly outlined. Rectangles whose match strength is equal to or exceeds the average match strength of the model will be outlined at intensity level 255. Rectangles with lower match strengths will be outlined at some arbitrary lower value that will be visually distinguished.

**Recommended Method:** The selected graph is traversed, and the updated positional information for each rectangle is used to draw its edges in the intensity image. Any order of traversal is acceptable, as is any standard method of drawing the lines in the image.

# 11. Required Timings and Instrumentation

Each participant in the benchmark exercise will be provided with a detailed set of reporting requirements for the data that is to be gathered during the exercise. Currently we intend to require timings for initialization, input, each task, any inter-task data transformations, result output, and total processing time. We will also specify timing requirements for some sub-tasks (kernel operations within tasks), and will encourage detailed reporting of any system-specific overhead times. We also plan to instrument the implementation process by requiring reports of code length, languages used, deviations from the recommended methods, hardware and/or simulator and/or estimation method used, and estimated programmer-hours for each task. We are also interested in the scalability of architectures, and will thus require some estimations of the effects of factors such as increased numbers of processors and improved technology.

# 12. Summary

Recent attempts at defining a vision benchmark include the Abingdon Cross problem, defined at the 1982 Multicomputer Workshop in Abingdon England, and the Tanque Verde benchmark suite, defined at the 1984 Multicomputer Workshop in Tucson Arizona. These benchmarks involved isolated processing tasks that were mostly concerned with low-level pixel processing. The most recent attempt at constructing a benchmark for vision emerged from the DARPA Image Understanding community, where a broad set of ten vision tasks were defined. The benchmark defined in this paper is an outgrowth of that benchmark, and is again sponsored by DARPA.

The purpose of this benchmark is to address the need for an integrated vision benchmark that transcends several different representations and forms of processing that are typical of complex vision applications. The scientific gain that should result from this exercise is a better understanding of vision architecture requirements, and the performance bottlenecks in different classes of machines, so that the needs of vision processing can be better addressed in the next generation of architectures.

This benchmark is not intended to be a challenging vision research exercise, and in fact it *should not* be. Instead, the intent is to exercise parallel architectures with a diverse range of operations that are commonly used and in a sequence that requires the sorts of transformations of data and control of processes that occur in a typical interpretation task. An attempt has also been made to incorporate many of the tasks from the first DARPA benchmark, in order to take advantage of the programming effort that has already been expended. The result is that, from an image understanding perspective, some aspects of the benchmark scenario may be viewed to be somewhat unrealistic and contrived as an interpretation task. Nevertheless as a benchmark, it is a valid means of testing performance.

A primary concern in designing a benchmark is that tasks must be as well-defined as possible so that the results from different machines will be comparable. The objective is to test the performance of the machines rather than the cleverness of their programmers. To this end, a recommended method has been specified for solving each of the tasks. The input data sets will be provided to those participating in the study so that all machines will be evaluated on the same test data.

This benchmark task suite involves recognizing an approximately specified 2 1/2D "mobile" sculpture composed of suspended rectangles, given images from intensity and range sensors. It is our intention that the test images be designed so that neither, by itself, is sufficient to form a complete match. The model of the object that is provided is approximate in the sense that the sizes, orientations, and depths of the rectangles as well as their spatial relationships are constrained to within some tolerances. A set of mobile models is provided, any one of which may be present in the images. The goal is to determine which of the models is actually present, the degree to which it is visible (matchable), and to update the model with positional data that has been extracted from the images.

Individuals and organizations that wish to participate in this benchmark exercise should contact the authors in order to obtain the most up-to-date version of this specification. The authors can also supply sample test data and source code for the sequential implementation of the benchmark suite.

## 13. Acknowledgements

This work was supported in part by the Advanced Research Projects Agency of the Department of Defense under contract number DACA76-86-C0015, monitored by the Engineer Topographic Laboratory.

We would like to thank Chris Brown, Claire Bono, C.H. Chien, Larry Davis, Todd Kushner, Ram Nevatia, Keith Price, George Reynolds, Lew Tucker, and Jon Webb for their many helpful comments and suggestions in response to the draft benchmark specification.

We would also like to thank Poornima Balasubramaniam, Sunit Bhalla, Chris Connoly, John Dolan, Martin Herbordt, Michael Scudder, Lance Williams, and especially James Burrill for their efforts in designing, programming, and debugging the sequential solution to the benchmark.

## 14. References

[Duff, 1986] Duff, M.J.B., "How Not to Benchmark Image Processors", in Evaluation of Multicomputers for Image Processing (Proceedings of the 1984 Multicomputer Workshop, Tucson, AZ), L. Uhr, K. Preston Jr., S. Levialdi, M.J.B. Duff editors, Academic Press, Orlando, FL, 1986, pp. 3-12.

[Preston, 1986] Preston, Jr., K., "Benchmark Results: The Abingdon Cross", in Evaluation of Multicomputers for Image Processing (Proceedings of the 1984 Multicomputer Workshop, Tucson, AZ), L. Uhr, K. Preston Jr., S. Levialdi, M.J.B. Duff editors, Academic Press, Orlando, FL, 1986, pp. 23-54.

[Rosenfeld, 1987] Rosenfeld, A.R., "A Report on the DARPA Image Understanding Architectures Workshop", Proceedings of the 1987 DARPA Image Understanding Workshop, February 1987, Los Angeles, CA, pp. 298-302, Morgan Kaufmann Publishers, Los Altos, CA, 1987.

[Uhr, 1986] Uhr, L., Preston Jr., K., Levialdi, S., Duff, M.J.B., "Preface", in Evaluation of Multicomputers for Image Processing (Proceedings of the 1984 Multicomputer Workshop, Tucson, AZ), L. Uhr, K. Preston Jr., S. Levialdi, M.J.B. Duff editors, Academic Press, Orlando, FL, 1986, pp. ix-xiv.

# Some Sample Algorithms for the Image Understanding Architecture

Charles C. Weems

Computer and Information Science
University of Massachusetts, Amherst, MA

**Abstract**

The Image Understanding Architecture is a heterogenous, multi-level, associative, parallel architecture that is designed to support real-time knowledge-based computer vision research and applications. A 1/64th scale proof-of-concept prototype is currently under construction by the University of Massachusetts and Hughes Research Laboratories. This paper briefly summarizes the IUA architecture and a programmer's model of the machine. Several algorithms are then presented that demonstrate some of the features of the low and intermediate levels of the architecture in performing computer vision tasks. These include both local and global associative operations, horizontal and vertical communication, and SIMD, multi-SIMD, and synchronous-MIMD processing modes.

## 1. Introduction

One goal of machine vision is the construction of a symbolic description of the environment depicted in an image. Such an interpretation involves not only labeling certain regions in an image, or locating a single object in the viewed scene, but often requires the construction of a three-dimensional model of the surroundings, with associated identification in the image of the two-dimensional projections of these three-dimensional models.

Because of the inherent ambiguities that are present in images of natural scenes, it is rarely possible to construct an interpretation directly from the pixel data. Additional knowledge must be used to reduce local ambiguity and to infer portions of objects that are missing in an image due to effects such as occlusion or shadows. Inference via stored knowledge and the reduction of ambiguity from "low-level" sensory processing are a part of what is referred to as "high-level" or knowledge-based vision processing. Without knowledge-based processing it would be impossible to interpret large portions of many images.

Since there is no simple computational transformation that will map arrays of sensory data onto the stored symbolic concepts represented in a knowledge base, it has become generally accepted that many levels of representation (data abstraction) and many stages of processing must take place to reliably interpret a scene. We will refer to the set of representations between the sensory data and the symbolic structures associated with objects in the environment as the intermediate level representation.

Image interpretation may thus be characterized as involving three different levels of processing, each with its own specific class of information. Additionally, those levels must be able to interact through bottom-up transformation of information and top-down control of processing. The low, intermediate, and high levels of representation and processing, together with our understanding of how those levels interact, provides the basis for the design of our Image Understanding Architecture (IUA). This paper begins with an overview of the IUA design, briefly discusses the programmer's model of the system, and then presents some algorithms and sketches of algorithms to provide the reader with a sense of the various modes in which the IUA can be used.

## 2. Overview of the Image Understanding Architecture (IUA)

The Image Understanding Architecture represents a hardware implementation of the three levels of abstraction embedded in our view of computer vision. It consists of three different, closely coupled parallel processors: the Content Addressable Array Parallel Processor (CAAPP)* at the low level, the Intermediate Communications Associative Processor (ICAP) at the intermediate level, and the

[1]This research was supported, in part, by the Advanced Research Projects Agency of the Department of Defense via Contract No. F49620-86-C0041 monitored by the Air Force Office of Scientific Research and under Contract No. DACA76-86-C0015 monitored by the Engineer Topographic Laboratory.

*The term "content-addressable" is a synonym for "associative" and is an alternate term that now is not as widely used as it was when some of our work began [Foster, 1976, Weems, 1984a]

Symbolic Processing Array (SPA) at the high level (Figure 1). The CAAPP and ICAP levels are controlled by a dedicated Array Control Unit (ACU) that takes its directions from the SPA level. In each layer of the IUA the processing elements are tuned to the computational granularity and algorithms required by that particular level of abstraction. For example, it is inappropriate to try to run concurrent LISP at the lowest level, because the low level is primarily concerned with fast pixel operations. Thus, the low-level processors are tuned for real-time image processing operations. At the highest level, on the other hand, symbolic AI processing will be the main objective, so the high-level processors are selected for their ability to run LISP code efficiently.

At the high level, the IUA is purely a MIMD parallel processor. Additionally, the intermediate and low levels of the IUA may be treated in a variety of modes of parallelism to allow multiple hypotheses from the SPA to be evaluated in parallel at the lower levels. These include the CAAPP operating in pure SIMD or multi-SIMD mode, and the ICAP operating in synchronous-MIMD or pure MIMD mode. A brief explanation of how the multi-SIMD and synchronous-MIMD modes differ from the familiar SIMD and MIMD modes is required. In multi-SIMD mode, the CAAPP cells execute in disjoint SIMD groups, with each group receiving a different instruction stream. This allows different algorithms to be run on disjoint portions of the image, with each algorithm performing pixel-parallel operations within its sector of the image. In synchronous-MIMD mode, the programming paradigm is more like SIMD than MIMD: rather than having completely different instruction streams executing concurrently, the ICAP processors execute similar instruction streams, (i.e. each ICAP can independently evaluate a conditional test and branch appropriately within a code segment tha all ICAP processors are executing in parallel) and globally synchronize for each stage of processing. Synchronous-MIMD has the advantage of being as simple to program as a SIMD system, but without the time penalty usually encountered in SIMD systems of having to sequentially execute all of the paths in a branching control structure.

### 2.1 The CAAPP (low) Level

The CAAPP is a $512 \times 512$ square grid array of custom 1-bit serial processors intended to perform low-level image processing tasks. The CAAPP is similar in many ways to CLIP-4 [Duff, 1978], MPP [Batcher, 1980], DAP [Hunt, 1981], GRID [Arvind, 1983], GAPP [Davis, 1984], and the Connection Machine [Hilis, 1986]. However, its architecture is especially oriented towards *associative processing* with an emphasis on hardware global summary feedback mechanisms. The CAAPP is also specifically designed to interact with the ICAP in a tightly coupled fashion for both bottom-up and top-down processing. Thus, the CAAPP has been

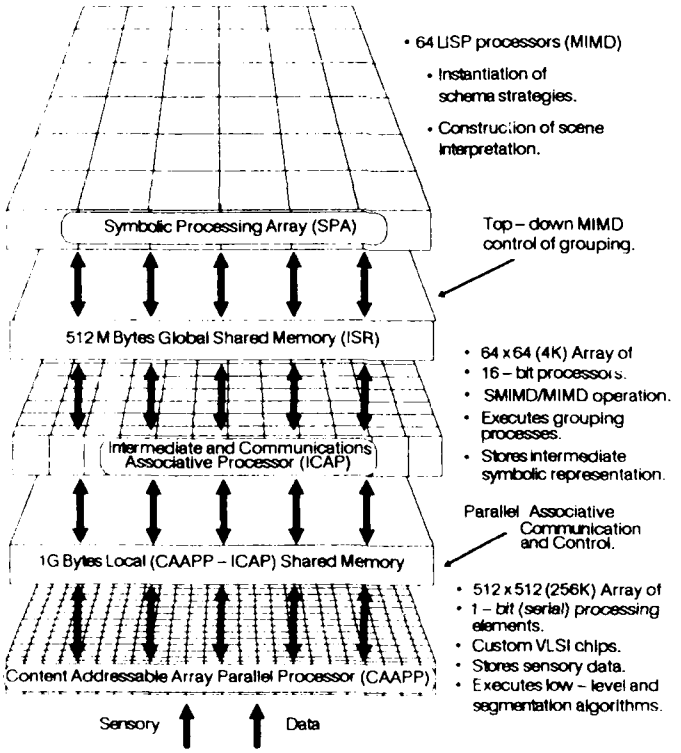



Figure 1: IUA Overview

tailored to permit flexible control, to provide rapid feedback to the controlling processes so that they may exercise control in response to actual image properties, and to integrate fully into a hierarchically organized vision architecture.

The CAAPP processing elements are linked through a four way (S,E,W,N) communications grid which is augmented with circuitry that allows certain types of long distance communication to take place quickly. Each processor can execute an instruction in 100 nanoseconds and contains 5 one-bit registers, an ALU data routing circuitry, and 320 bits of RAM that acts as an explicitly managed cache memory. Each element has access to a 32K-bit backing store memory that is dual-ported with the ICAP. The backing store is also referred to as the CAAPP-ICAP shared memory (CISM).

The key to integrating the CAAPP into the IUA is its combination of associative feedback and control mechanisms. The principle feedback mechanism in the CAAPP is the array-wide logical OR output, called Some/None, which indicates whether any CAAPP cells are in a given state represented by the response bit. At the end of each instruction cycle the global controller receives a Some/None signal for the full array, while the ICAP processors receive the Some/None indication for that portion of the CAAPP array connected to each of them.

A count of all responding cells is also available at the global controller. The counting operation is used to gather statistics about an image and the results of processing. For example, through counting we may quickly determine the mean and standard deviation of an attribute value for a given set of processors (see section 4.4). Each ICAP processor receives the count for the $8 \times 8$ subarray of the CAAPP associated with it.

Communication among CAAPP cells may take place in four different ways. One way is through global feedback and rebroadcast. This method is used when all or most of the CAAPP processors must be told the value of one of the processors (e.g. broadcasting the maximum value so that all cells can normalize their values). A second way is via the ICAP. In some cases it is more efficient to transfer CAAPP data to the backing store and let the ICAP move it across the array and place it in the backing store of the appropriate CAAPP cell. The third way uses the nearest neighborhood (S,E,W,N) mesh, which allows a CAAPP processor to read a bit from up to two of its neighbors at once. This is similar to the network employed in other mesh-connected SIMD parallel processors. The remaining communication mechanism is described in the next section.

### 2.2 The Coterie Network

The fourth means of communication among CAAPP processors involves a new and powerful variation on the nearest neighbor mesh called the Coterie network. This is similar to the reconfigurable buses proposed by Kumar [Kumar, 1987], Miller and Stout [Stout, 1986] and the polymorphic torus proposed by Li [Li, 1987], but differs in that it allows general reconfiguration of the mesh, and multiple processors may write at once. By adding the simple switch network shown in Figure 2, it is possible, under program control, to create independent groups of processors that share a local associative Some/None feedback circuit. The isolated groups of processors can then respond to globally broadcast instructions in a locally data-dependent fashion, which permits parallelism to be employed with more flexibility. For example, suppose that an image is divided into a large number of regions, and that we wish to determine

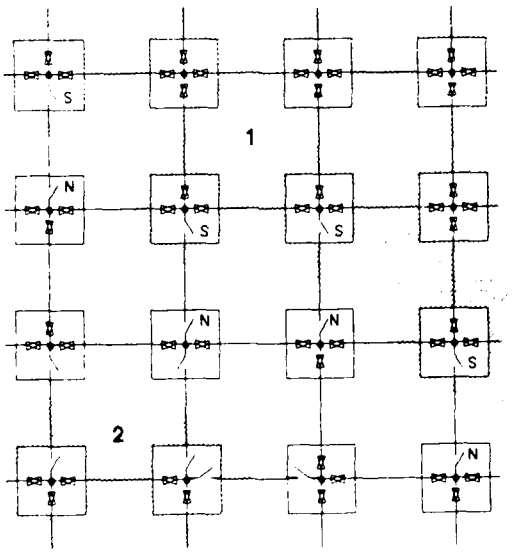



Figure 2: The Coterie Network

some attribute for each of the regions. In a typical SIMD architecture this would be done by sequentially selecting each region for analysis or in parallel by complex communication between neighbors where the attribute is computed via a propagating wave that checks region labels at each step. In the CAAPP, however, all regions can, in many cases, perform their own local evaluation in parallel without having to check region labels after one initial step of neighbor comparison.

The name Coterie Network is based upon the similarity of the isolated processor groups to a "coterie": that is, a group of people who associate closely because of common purposes, interests, etc. [Random, 1987]. The isolated groups of processors are thus referred to as coteries. Note that the Coterie Network is separate from the nearest neighbor mesh, which we refer to as the SEWN Mesh.

Creation of a set of coteries typically begins with opening all of the switches that link processors. Using the SEWN Mesh, the processors compare their own values with the values of their neighbors. They then close the switches that connect them to neighbors with similar properties, leaving open the switches that would connect them to dissimilar neighbors. Of course similarity can be defined by an operation such as a global broadcast of a threshold and a local comparison. In this way, processors with similar properties establish independent coteries. It should be fairly obvious to the reader that, among other things, each region of a segmentation could be a coterie of cells. Because the CAAPP processors can save and restore the switch settings that make up a set of coteries, it is possible to reconfigure the Coterie Network from one processor interconnection pattern to another by broadcasting a single instruction.

Within a coterie, there is a mesh of wire that all of the processors are connected to. The ACU may instruct each active CAAPP processor to output a bit onto its coterie's mesh and then read whatever bit value is currently on the mesh within its coterie. When more than one processor in a coterie tries to output a bit onto the mesh, the value that appears on the wire is the logical OR of the output bits of all of the processors in the coterie. The shared mesh is thus functionally equivalent to the global Some/None feedback circuit except that its output is only locally formed and available within a coterie. In addition to its associative feedback function, the coterie mesh can be used to broadcast a value from a single processor to every member of the coterie as in a Broadcast Protocol Multiprocessor [Levitan, 1984]. This is accomplished by first selecting a single processor within each coterie, using an associative search operation in parallel within all coteries. Subsequent ACU instructions for placing a value onto the mesh will only be performed by these selected cells. However, all of the cells will perform the operations for reading the value that is on the mesh. In this way, the coterie network can be used for local broadcasting of data values. The local feedback and broadcast processes can occur in every coterie in parallel.

### 2.3 Inter-level Communication Between the CAAPP and ICAP

The principal mechanism for transferring data between the CAAPP and ICAP is the CISM (or backing store). Each ICAP processor has access to the 256K-byte block of memory that also acts as the 32K-bit backing store for each of the 64 CAAPP cells associated with an ICAP processor. Swapping between the CAAPP and CISM is accomplished by dual-porting a portion of the on-chip CAAPP memory. When data is moved between the CAAPP and the CISM it goes through an automatic corner turning mechanism that provides bit-serial data access to the CAAPP and byte-parallel access to the ICAP.

### 2.4 The ICAP (Intermediate) Level

The ICAP is designed to manipulate tokens (symbolic descriptions of extracted image events and their associated attributes) at the intermediate level and to support data base functions that allow access to these tokens by grouping processes running on the ICAP and by symbolic interpretation processes, running on the SPA processors. For example, the recognition of a house roof in an image may require the ICAP to group together long, straight, parallel lines, and then to extract parallelograms that are candidate roof outlines. Should the need arise, the results of further processing in the CAAPP can be integrated with the representation in the ICAP because the ICAP representation is in approximate registration with the original image events in the CAAPP.

The ICAP is a square grid (64 × 64) array of Texas Instruments processing TMS320C25 16-bit digital signal processor chips. Each of the 4096 ICAP processors consists of a CPU, 256K bytes of local RAM, 384K bytes of dual ported memory for interacting with the CAAPP and SPA, and network communications hardware. The ICAP processors operate at 5 million instructions per second and can perform a 16-bit multiply and accumulate operation in a single instruction time. In addition to its speed, a digital signal processor was chosen at the ICAP level because its instruction set and arithmetic capabilities are well suited for performing computations in spatial geometry. Three-dimensional geometric projections, and computing distances, and matching operations are among the more frequently employed operations at the intermediate level of vision processing.

Control of the ICAP is provided by the ACU (in Synchronous-MIMD mode) and by the SPA (in MIMD mode). Once sensory events have been extracted and represented symbolically at the intermediate level in the ICAP (and continue to evolve as grouping operations take place), each of the SPA processors may then query the ICAP in parallel to establish and verify hypotheses. The ICAP provides three different global OR responses that are used by the controller to determine the status of processing in the

ICAP array. The choice of meaning for each signal is left up to the programmer. For example, the programmer may choose to have them indicate completion of a task in the ICAP array with or without exceptions. Another use is as an associative Some/None mechanism. A global summation mechanism is also provided that uses the global count hardware in the CAAPP to form a sum of an 8-bit value from each ICAP processor.

The horizontal links between the ICAP processors provide the intra-level communications necessary for grouping and merging processes to operate on token attributes and token relations within the intermediate symbolic representation. In the IUA prototype, which has only 64 ICAP processors, the bit-serial I/O links between the processors are connected through a centrally controlled 64 × 64 bit-serial crossbar switch. Various methods of extending the ICAP communications network to the full-size ICAP array are under consideration.

### 2.5 Inter-level Communication Between the ICAP and SPA

The ICAP-SPA Shared Memory (ISSM) provides the principal communication path between the top two levels of the IUA. It is viewed as an I/O device by each ICAP processor. A given ICAP processor can write (or read) values to (from) an I/O buffer in the ISSM. The ICAP then initiates a block transfer between the I/O buffer and a page of its choosing in the ISSM RAM. An ICAP processor may only access the 128 K-byte segment of ISSM that is associated with it. However, each SPA processor has global access to the entire ISSM for all of the ICAP processors. This structure allows processes in the SPA to access the results of ICAP processing regardless of their spatial locations in the array.

### 2.6 The SPA: High Level Processing

The SPA processors will run a LISP-based blackboard system [Erman, 1980, Nii, 1986, Draper, 1987a,b,] in which various knowledge-based processes will cooperate and compete in the generation and verification of hypotheses about the content of the image and its relationship to models of the environment. From the point of view of the blackboard system, the CAAPP and ICAP will appear as knowledge sources at different levels of abstraction. Knowledge-based processes in the system can activate different processes in the CAAPP and ICAP either for the full array or for independent sub-arrays. Thus, the SPA processors operate in MIMD mode with communication through the blackboard. The detailed architecture of the SPA has not yet been fully defined. In the first prototype of the IUA, which is a 1/64th vertical slice of the full IUA, the SPA will be a single Motorola M68020 class processor. A separate research investigation within the UMass VISIONS project is currently exploring the implementation of cooperative algorithms and data structures using a shared-memory multiprocessor at the SPA level [Draper, 1987]. This experience is providing additional direction to the future scaling up of the IUA at the SPA level.

Currently, the full SPA is envisioned as consisting of 64 or more processors, each capable of running LISP. Each processor will have some local memory and will have access to a global shared memory that will include the ISSM, and the blackboard. The shared memory decouples the SPA processors from the locality of information in the image.

### 2.7 The ACU: Controlling the CAAPP and ICAP

One major design goal for the ACU was to maximize the rate at which instructions are issued to the CAAPP. This meant that the overhead for controlling loops, branches, and subroutine calls in the ACU had to be minimized. A second major design goal for the ACU was to minimize the cost of implementing a complete development environment for it. Preferably, the ACU would execute a commonly used instruction set so that software could be transported from an existing machine.

Clearly, the first goal required a custom processor, while the second goal dictated an off-the-shelf processor. The solution to this dilemma was to incorporate both into the ACU design. Thus, the ACU contains two separate processors that can issue instructions to the CAAPP (and control the ICAP as described below). The two processors are referred to as the Macro-controller and the Micro-controller.

The Macro-controller is a Motorola M68020-based system that brings with it the wide range of software tools that are available for that processor. It can issue instructions to the CAAPP in two ways. The simplest way is to take direct control of the instruction bus and write out data values that will be interpreted as instructions by the processor arrays. Even at its maximum rate, however, the 68020 can only issue instructions at about one-tenth of the rate that the CAAPP can execute them. The second method for the Macro-controller to issue instructions is to issue subroutine calls to the Micro-controller.

The Micro-controller is a custom processor, driven by horizontal microcode. It is capable of issuing an instruction to the CAAPP every 100 nanoseconds, with minimal overhead for loop, branch, and subroutine control. The Micro-controller will have a large library of CAAPP routines in its program memory, any of which can be called by the Macro-controller. When the Micro-controller completes execution of a CAAPP routine, it returns a status flag to the Macro-controller which may then issue a new call.

The routine-calling mechanism permits the user to write applications in a high-level language for the Macro-controller, and yet obtain good peak instruction rates for operations on the CAAPP. Although this does not provide 100% utilization of the CAAPP, it is reasonable to expect 50%

utilization in many cases, which should be adequate for most research and development situations.

Although the only source of instructions for the CAAPP is the ACU, the ICAP processors each have their own program memory. The ICAP program memory is loaded with a large library of service routines upon system initialization. The way in which the ACU issues instructions to the ICAP is by storing a user program into ICAP program memory and then issuing an interrupt to the ICAP that causes it to jump to the user program. (The program is broadcast to all of the ICAP program memories in parallel.) An ICAP user program is typically just an execution script (written in C, Forth, or assembly language) of calls to the ICAP library. Thus, the ACU and ICAP interact very little when a program is running in the ICAP; the exception is when the ICAP program reaches a global synchronization point: that must be mediated by the ACU. The ACU can also set the ICAP to operate in MIMD mode, by turning control over to a task queuing program in the ICAP processors. The queuing program reads execution scripts from the ISSM according to a predefined protocol. When the ICAP is executing in MIMD mode, it depends upon the SPA to provide coordination of any required synchronization of ICAP processors.

The ACU thus supports for the close interaction between the CAAPP and ICAP during the initial phases of interpreting an image. However, the ACU also permits the CAAPP and ICAP to work independently, with the ICAP taking directions from the SPA as the high level interpretation processes come into play. This allows the CAAPP to concurrently perform additional-low level processing, such as integrating information from other sensors or starting to process the next image.

## 3. IUA Programmer's Model

Just as the IUA is a hierarchical architecture, the IUA programming model is also hierarchical. At the lowest level, a programmer may write assembly language primitives for the CAAPP, ICAP, and SPA. More typical, however, will be high-level programs for the SPA. Between these two extremes are programs for the ACU that control the CAAPP and ICAP arrays in cooperation with the SPA.

SPA programs are written for a shared-memory multiprocessor model, typically using a high level language with extensions to support concurrency. Because the SPA processors are all identical, any process may execute on any processor and have access to the entire intermediate level database (ISSM) and the blackboard. The ACU is interfaced to the SPA in such a way that it acts as a special SPA processor that has control over the CAAPP and ICAP arrays. Thus, ACU programs communicate with SPA processes via the same mechanisms that SPA processes use to communicate with each other.

To the application developer, the IUA is simply programmed with a set of concurrent processes, some of which perform high-level vision tasks and some of which service requests for CAAPP and ICAP processing. The server processes run in the ACU and view the CAAPP and ICAP as a pair of attached array processors.

The software interface to the CAAPP and ICAP takes two forms. An ACU program will usually just call library subroutines that cause the arrays to perform predefined operations. However, if an operation is required that isn't in the library, then a new subroutine must be written. CAAPP routines can be written inline as part of an ACU program via a series of calls to a subroutine that issues individual CAAPP instructions. CAAPP routines can also be written in assembly language, or a high level language and compiled as linkable modules. ICAP routines must be precompiled and downloaded to the ICAP program memory because the function of the ACU is primarily to issue SIMD instructions to the CAAPP. The ACU merely coordinates execution of routines that are already stored in the ICAP processors whenever they are operating in synchronous-MIMD mode, or interacting closely with the CAAPP.

When an application requires maximum performance from the CAAPP and ICAP, it must be micro-coded for execution on the micro-controller. The micro-controller is designed to be a complete processor, capable of executing general purpose programs. Thus, for real-time applications, a typical development scenario would involve rapid prototyping of an implementation on the Macro-controller, followed by migration of the high-level code into Micro-controller instructions. In the future, tools and compilers will be developed for the Micro-controller that will aid the code migration processes. However, it is reasonable to assume that the Macro-controller development environment will always be more attractive. Thus, it is expected that the two-stage development process will remain as the standard approach to implementing real-time applications on the IUA.

The IUA programming environment currently exists as a set of software simulators. The ACU, CAAPP, and backing store portions of the simulator are available on VAX and SUN systems, while the full IUA simulator is running on an Explorer LISP workstation augmented by a TI Odyssey parallel signal co-processor board containing four TMS32020 processors. The simulators can be programmed in a variety of languages. LISP, Forth C, etc., may all be used to write ACU programs that drive the CAAPP. The ICAP processors, however, are currently programmable only in Forth and assembly language (a C compiler is under development by TI). The simulated SPA may be programmed in LISP or Prolog.

## 4. Sample Algorithms

While the purpose of this section is to provide a sense of

the types and and range of operations that can take place on the IUA, it is by no means a complete discussion of all of the system's capabilities. The algorithms presented here are specifically intended to demonstrate the various forms of communication that occur within and between the CAAPP and ICAP levels. Processing and communication that involves the SPA level will be left for future presentation.

This section will begin with several fairly simple but detailed algorithms in order to show how the IUA is programmed. The algorithmic notation used is very close to one of the programming languages employed with the IUA software simulators. However, the notation is simplified to improve the clarity of the presentation. Macro operations are also used where their machine language implementation is not an important element of the algorithmic method. For example, adding two 8-bit quantities in the CAAPP is actually performed bit-serially by a sequence of instructions, but a typical program will make use of the standard macro for addition to perform this operation. Following the detailed algorithms, several more algorithms will be sketched whose complexity makes them too lengthy to be presented here in great detail. The concepts behind the algorithms are worth considering, however, because they demonstrate additional capabilities of the IUA.

### 4.1 Select Greatest Responding Value

This algorithm (Figure 3) demonstrates the use of the associative Some/None feedback from the CAAPP array. The goal is to select, from among all active cells, the cell or cells that have the greatest value in a given field of their memory. In addition, that value is to be made available in the ACU for subsequent processing.

The algorithm begins by loading the high order bit of a given field into the response register of all active cells. The global controller then tests the Some/None output of the array. If any cells have their high order bit set, then they are candidates for the maximum value; in which case, any cells that have a zero in their high order bit are then deactivated. However, if no cells have their high order bit set, then none are deactivated because they are all still potential candidates. This process repeats with each successively lower order bit in the field. When the low order bit has been processed, only those cells that contain the maximum value will remain active. For each iteration, the controller saves the Some/None response so that the maximum value is available in the controller at the conclusion of processing. This takes 24 CAAPP instruction cycles (2.4 microseconds) for an 8-bit value.

### 4.2 Label Connected Components

The local associative Some/None operation provided by the Coterie Network is demonstrated by this algorithm. Because finding a maximum uses only broadcast and Some/None feedback, it can be performed locally within a coterie and in parallel with every other coterie. This leads to the simple algorithm for computing a connected component labeling of an image shown in Figure 4.

The algorithm begins by loading each processor with its address in the array from the backing store memory, which serves to give each processor a unique number. Next, the Coterie Network switches are opened between processors that are on region boundaries (i.e. between pairs of processors that have different values) establishing a coterie for each image region. Lastly, all regions in parallel determine their local maximum address value. Note this is the same algorithm as for finding a maximum value in the entire array except that the coterie Some/None response is used in place of the global Some/None response to control the setting of activity. As part of finding the maximum, every processor in a coterie stores the maximum address value for all cells in its coterie in its own memory. Because this value is different for every region, the result is that each connected group of processors is assigned a unique label that is common to every processor within a group. From our electrical simulations of the Coterie Network, we calculate that this algorithm will take approximately 50 microseconds to execute.

### 4.3 Histogram

The preceding algorithms use only the Some/None Response form of feedback. The response count is of equal importance in many of our algorithms. For example, we can form a histogram of any numerical feature in the CAAPP using the response count. This is quite simple to do: For each bucket in the histogram we associatively select those cells whose values fall within the range of the bucket by broadcasting the minimum and maximum value of the range, comparing with the cell's value, and appropriately setting the response register to 0 or 1; then a count of the

```
- - Beginning with the high-order bit
FOR Bit_:= Field_Length - 1 DOWN TO 0 DO
      Response:= Field[Bit_Num]                    {Put bit in response register}
      IF Some                                      {If any cell has a 1 in this bit}
          THEN
              Activity:= Response                  {Then turn off activity in cells with a 0 in this bit}
```

Figure 3: Finding a Maximum Value in the CAAPP

```
Load_Processor_Addresses

Coterie_Switches:=(Open, Open, Open, Open)                    {Initialize coterie switches}
FOR Neighbor:=North TO West DO                                {Initialize flag for each neighbor}
    Equal:=True
    FOR Bit_Num:=Field_Length -1 DOWN TO 0 DO                 {For each bit in field}
        Equal:=Equal AND (Neighbor.Field[BitNum]=Field[Bit_Num])   {Compare own bit with neighbor}
    IF Equal                                                  {If field value matches neighbor, bit for bit}
        THEN                                                  {Then close the coterie switch to connect with that neighbor}
            Coterie_Switch[Neighbor]:=Closed
FOR Bit_Num:=Address_Length -1 DOWN TO 0 DO
                                                              {Find maximum addresses in coteries}
    Response:=Address[Bit_Num]                                {Put bit in response register}
    IF Coterie_Some                                           {If any cell has a 1 in this bit}
        THEN
            Activity:=Response                                {Then turn off activity in cells with a 0 in this bit}
    Component_Label[Bit_Num]:=Coterie_Some!                   {Save bit values for component label}
```

Figure 4. Connected Component Labelling using the Coterie Network

responding cells gives the histogram bucket value. The time to form the histogram is thus proportional to the number of buckets in the histogram (typically about 1.6 microseconds per bucket) and is independent of the number of values in the array.

### 4.4 Compute Average Value

Figure 5 gives a CAAPP algorithm that uses the response count to compute the mean of the values stored in selected cells. The algorithm begins by summing the values in the selected cells. Starting with the high order bit position of the values to be summed, each bit of the values in the selected cells is separately counted. The counts are each added to the overall sum after being appropriately scaled by a power of two. The algorithm concludes by setting each cell's response bit equal to its activity bit so that the response count will be the number of active cells, and dividing the sum by that count to get the mean of the values.

### 4.5 The Sobel Edge Operator

Of course, in addition to processing that is oriented around associative feedback, the CAAPP is able to perform the usual image processing and low level vision algorithms that do not depend upon feedback to the controller. For example, smoothing operators such as Gaussian convolution, edge detectors such as the Sobel and Canny operators, local pixel comparisons, line curvature, border following, etc. all use the mesh connected operations that are typical of this class of machines.

To demonstrate communications between neighboring CAAPP PEs, the algorithm for performing a Sobel operation is shown in Figure 6. The Sobel computes the local X and Y gradient magnitude at each pixel in an image. These X and Y magnitude vectors subsequently can be combined to form the orientation and magnitude of the local gradient at each pixel. Pixels with large gradient magnitudes are frequently associated with strong edges or lines in an image, and are thus likely to be of use in interpreting an image.

The Sobel operator requires that the image be convolved with two different 3 x 3 masks. One mask computes the gradient magnitude in the X direction while the other computes the magnitude in the Y direction.

```
Sum:=0                                                        {Initialize sum}
FOR Bit_Num:=High_Order DOWN TO Low_Order DO    {Count each bit in field and add to sum, scaling appropriately}
    Response:=Field[Bit_Num]
    Sum:=Sum*2+Response_Count
Response:=Activity                                            {Count number of active cells}
Mean:=Sum/Response_Count                                      {and compute mean}
```

Figure 5. Computing the Mean of Values in Selected CAAPP Cells Using the Response Count Operation

The masks are:

| X magnitude: | Y magnitude: |
|---|---|
| -1 0 1 | 1 2 1 |
| -2 0 2 | 0 0 0 |
| -1 0 1 | -1 2 -1 |

In the CAAPP, the X magnitude is computed by first having each cell double its own value, and then add the values of its North and South neighbors. This intermediate result is then used to compute the actual X magnitude by subtracting the intermediate value of each cell's West neighbor from the intermediate value of its East neighbor. A similar sequence of operations is used to compute the Y magnitude. The algorithm is shown in Figure 6:

```
{Compute X Magnitude}
   Double_Own := Own_Value + Own_Value
   Int_Result := Double_Own + North(Own_Value)
   Int_Result := Int_Result + South(Own_Value)
   X Magnitude := East(Int Result)
   X_Magnitude := X_Magnitude - West(Int_Result)
{Compute Y Magnitude}
   Int_Result := Double_Own + East(Own_Value)
   Int_Result := Int_Result + West(Own_Value)
   Y_Magnitude := North(Int Result)
   Y_Magnitude := Y_Magnitude - South(Int_Result)
```

Figure 6. Sobel Algorithm
for the CAAPP

Assuming that the operation is being applied to 8-bit integer pixels, it would require 100 CAAPP instruction cycles (10 microseconds) to compute the components of the Sobel operator for the image. The gradient magnitude and orientation can be computed from these components by applying the standard formulas as a sequence of CAAPP arithmetic operations.

### 4.6 Create Border Corner Lists

In addition to performing associative Some/None operations, the Coterie network may be used to pass messages across the mesh by providing direct links between non-adjacent processors. In the algorithm shown in Figure 7, it is assumed that some corner detection operation has been performed on the borders of regions in the image. The result is a sparse set of processors that are labelled as corners (ie. those processors whose Corner_Tag field is set to true). One useful feature that can be extracted for each region is a list of the positions of its corners. The following algorithm forms the corner lists for all regions in parallel. It begins by making each region an independent coterie, using the connected components labelling algorithm presented earlier. A single cell in each coterie is selected as the coterie leader. In this case, the chosen cell is the member of the coterie whose cell address equals the region's component label. The leader is responsible for collecting the corners for its region, and passing them to the ICAP which stores them in a list.

The first corner is determined by selecting the corner cell in each coterie with the maximum address. As part of this process, the coterie leader learns that cell's address and passes it to the ICAP processor associated with the CAAPP chip containing the leader. The selected corner cell is then shut off and the process is repeated so that the next corner is selected. The loop ends when there are no more corners to select, at which point every corner will have been passed to the ICAP by its coterie leader.

This algorithm causes the corner lists to be created in reverse raster-scan order, which is adequate if the regions are simple convex figures. However, for more complex regions, it may be difficult to reconstruct the shape of a region given a corner list in this order. A better arrangement is to list the corners in clockwise (or counterclockwise) boundary traversal order (ie. the order in which corners would be encountered as the cells at the boundary of the region are traversed, starting from some arbitrary boundary point). The coterie network can also be used to accomplish this task. Although the basic concept for doing this is quite simple, in practice it is complicated by considerations of regions that are one pixel wide and regions that completely enclose other regions. Because the algorithm is more complex, it is only discussed here in general terms; the detailed discussion will be left to a future paper.

As in the preceding algorithm, the first step is to label connected components. After connected component labelling has been performed, each member of a component examines its neighborhood to determine whether any neighboring cell has a different component label. Any cell that has a neighbor belonging to a different region is at the boundary of its own region. In the simplest case, the cells that are at a region's boundary form a chain that are at a region's boundary for a chain that is a one-pixel wide closed loop. Each cell in the chain will have two neighbors; one in the clockwise direction and the other in the counterclockwise direction around the loop. The cells that make up a boundary chain can then set their coterie switches so that the chain becomes a separate coterie. (Some of the complexity of the actual algorithm stems from the situations in which a boundary chain is not a simple closed loop and how the different cases are handled.)

A leader is selected for each of the boundary-chain coteries. Each leader opens the coterie switch connecting it to

```
Label_Connected_Components        {Each component is given a unique label that is equal to the maximum cell address
                                  within the component. The label is stored in a field called Component_Label in
                                  each cell                                                                    }
Start ICAP(Corner List_Builder)   {A process is started in the ICAP that will respond to each Signal ICAP operation
                                  (except the first) by picking up a corner from the backing store, and adding it to
                                  the list for the appropriate component region                                }
Activity := 1!                                                          {Turn on all cells}
Leader := Address = Component_Label                                     {Identify the leader for each coterie}
Backing Store Write(Leader)                                             {Copy leader tags to backing store}
Response := Leader
Latch_Local_Count                                                       {Count of leaders in each CAAPP chip}
Signal ICAP                       {The ICAP processor associated with each CAAPP chip reads the local count to
                                  determine the number of coteries for which it will collect corners. Each ICAP
                                  processor also scans its portion of the backing store to determine the addresses of
                                  the leaders in the coteries associated with it                               }
Activity & Response := Corner_Tag                                       {Activate all corner cells}
WHILE Some DO                                                           {Select corner with greatest address in each coterie}
    FOR Bit Num := Address Length - 1 DOWNTO 0 DO
        Response := Address[Bit_Num]
        Next Corner[Bit_Num] := Coterie_Some!   {Store each bit of the greatest address in all members of the
                                                coterie (including the leader) by ignoring the activity bit  }
        IF Next Corner[Bit Num] THEN
            Activity := Response                                        {Turn off cells that are less than the max}
    Corner_Tag := False                                                 {Disable the greatest corner once it's found}
    Backing Store Write(Next_Corner)                                    {Copy the address to the backing store}
    Signal ICAP                   {The ICAP processor associated with each CAAPP chip picks
                                  up the address stored in each coterie leader location in its por-
                                  tion of the backing store and saves it in a list             }
    Activity & Response := Corner_Tag!                                  {Activate remaining corners}
END WHILE
```

Figure 7: Extracting Border Corner Lists

its counterclockwise neighbor in the loop. The loop is thus transformed into an open figure with the leader at one end. Every cell in the chain that is also tagged as a corner now opens the switch connecting it to its clockwise neighbor in the chain. Next, each leader broadcasts a bit to its coterie. Because the corner cells have broken the coterie, the bit will only reach the first corner clockwise along the boundary from the leader. That corner is then activated and transmits its address back along the coterie to the leader which subsequently passes the address to the ICAP. The active corner then closes the switch to its clockwise neighbor and deactivates itself so that further broadcasts from the leader will pass through it. The process is repeated until all corner cells have been read out to the ICAP.

### 4.7 Region Adjacency Graph

A similar algorithm involves collecting a list of adjacent region labels for each region in an image. This algorithm begins by having every boundary cell get the label of its neighbor that is in another region, using the SEWN mesh. Each region then performs a coterie-select-greatest operation on these region labels, and a region label is output to the ICAP via the coterie leader. All boundary cells that have the selected label are then shut off and the test is repeated to obtain the next region label. The process is complete when there are no more labels to output. Because a region label is the address of the coterie leader for a region, and the ICAP processors are spatially collocated with respect to the CAAPP cells, each ICAP processor can directly compute the ID number of the other ICAP processors that contain descriptions of adjacent regions.

### 4.8 Rule-Based Region Merging

Once the ICAP processors have collected the information required to describe a specific type of image event, for example lines or regions, the ACU can broadcast rules (or constraints) to the ICAP that cause it to take some action. Given that the ICAP contains an attribute list for each region consisting of its size, average intensity, list of border corners, and list of adjacent regions, the ACU could broad-

cast a rule to the ICAP that says "If a region is below X in size, and is adjacent to one or more regions that exceed size Y, it should be merged with the adjacent region whose intensity differs least from its own, but only if the intensity difference is less than threshold Z."

Such a compound rule will actually take the form of a processing script that is downloaded to the ICAP processors from the ACU via a broadcast to the ICAP program memories. The script will actually be a series of calls to library routines that have been pre-stored in the ICAP. In this case, the script would select regions larger in size than threshold Y to transmit their size, intensity, and ID to all ICAP processors that contain regions on their list of adjacent regions. The ICAP processors then compare each region that is smaller than X to the size and intensity of each region for which information was received. If the condition for a merge is met, then an ICAP processor transmits all of the information for the smaller region to the processor that is responsible for the larger region. The processor that contains the larger region adds the smaller region's information to its database. The processor that contains the smaller region transmits the new region label to the CAAPP by writing the label into the backing store for the region's coterie leader. The ACU then instructs all coterie leaders that have received new labels to broadcast the new label to their coterie and then resign as coterie leader. The cells that are on the common boundary between the two regions then close their coterie switches so that the two coteries are merged into one. The ICAP processor that was responsible for the smaller region then deletes the region's information from its database.

A variety of interprocessor communication topologies can be supported in the ICAP, due to the flexibility of the centrally controlled network switch. For example, topologies such as the mesh, ring, hypercube, or shuffle can be built with the ICAP switch. In this algorithm the mesh topology would be used because the communication tends to be local in nature. (Information that is to be sent between non-adjacent ICAP processors is relayed by those processors that are between them.) In order for an ICAP processor to communicate with its four neighbors, it sends to one neighbor while receiving from the neighbor in the opposite direction. Each time the ICAP is ready to switch directions, it signals the ACU which changes the ICAP connection pattern and resynchronizes the serial ports on all of the ICAP processors. Thus, all ICAP processors might initially transmit to the North and receive from the South. When all transmissions to the North are finished, the ACU changes the connection pattern so that all messages will be transmitted to the West and received from the East, and then signals the ICAP processors to start transmitting again.

## Summary

The Image Understanding Architecture is a three-level parallel processor that is designed to support the multiple levels of representation and abstraction inherent in our view of knowledge-based computer vision processing. Each level is designed to perform a suite of tasks most appropriate for that level of abstraction.

Just as the IUA is a hierarchical system, the programmer's model of the IUA is also hierarchical. At the lowest level, a programmer may write assembly processors support routines for the CAAPP, ICAP, and SPA (CAAPP support routines are actually written in microcode for the Micro-controller portion of the ACU). At the intermediate level of the programmer's model are applications written for the Macro-controller portion of the ACU. The highest level of the model, where the majority of applications are written, takes the form of concurrent processes that execute in the SPA using a shared-memory multiprocessing model of computation.

A collection of algorithms has been presented that demonstrates some of the ways that the CAAPP and ICAP levels of the IUA may be programmed. In particular, processing modes have been demonstrated that use global associative processing; local associative processing via the Coterie Network; horizontal communication via the SEWN mesh, coterie network, and ICAP centrally switched network; vertical communication via the local count and CISM; and the SIMD, Multi-SIMD, and Synchronous-MIMD forms of parallelism.

The IUA currently exists as a set of software simulators. However, the custom CAAPP chips are currently being fabricated and a CAAPP-CISM-ICAP-ISSM test structure has been breadboarded. The prototype IUA system is scheduled for completion in the Fall of 1988.

## Acknowledgements

We thank our colleagues at the University of Massachusetts, Dr. Edward Riseman and Dr. Allen Hanson; at the University of Pittsburgh, Dr. Steven Levitan; and at Hughes Research Laboratories, Dr. David Shu and Dr. J. Gregory Nash, for all of their contributions to this project. We would also like to thank Dr. Caxton C. Foster for his long-term contribution of ideas in content addressable memories and processing that provide a foundation to this work.

## References

[Arvind, 1983] Arvind, D.K., Robinson, I.N., and Parker, I.N., A VLSI Chip for Real-Time Image Processing, Proc. IEEE International Symposium on Circuits and Systems, 1983, pp. 405-408.

[Batcher, 1980] Batcher, K. E., Design of a Massively Parallel Processor, *IEEE Trans. Comp.*,29:1-9, September 1980.

[Davis, 1984] Davis, R., Thomas, D., Geometric Arithmetic Parallel Processor-Systolic Array Chip Meets the Demands of Heavy-Duty Processing, Electronic Design, October 31, 1984, pp. 207-218.

[Draper, 1987a] Draper, B.A., Collins, R.T., Brolio, J., Griffith, J., Hanson, A.R., Riseman, E.M., "Tools and Experiments in the Knowledge- Directed Interpretation of Road Scenes", Image Understanding Workshop Proceedings, Morgan Kaufmann, Los Altos, CA, 1987.

[Draper, 1987b] Draper, B.A., Collins, R.T., Brolio, J., Griffith, J., Hanson, A.R., Riseman, E.M., The Schema System: Knowledge Based Vision (in preparation).

[Duff, 1978] Duff, M.J.B., Review of the CLIP Image Processing System, Processings of the National Computer Conference, 1978, AFIPA, pp. 1055-1060.

[Erman, 1980] Erman, L., et al., The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty, *Computing Surveys*, 12:213-253, 1980.

[Foster, 1976] Foster, C. C., *Content Addressable Parallel Processors*, New York: Van Nostrand Reinhold, 1976. (a)

[Hanson, 1986] Hanson, A. R., Riseman, E. M., A Methodology for the Development of General Knowledge-Based Vision Systems, (to appear). In: *Vision, Brain, and Cooperative Computation*, M. Arbib and A. Hanson (Eds.), Cambridge, MA: MIT Press, 1986.

[Hillis, 1986] Hillis, D.W., The Connection Machine, MIT Press, Cambridge, 1986.

[Hunt, 1981] Hunt, D.J., The ICL DAP and its Application to Image Processing, in Languages and Architectures for Image Processors (M.J.B. Duff, S. Levialdi eds.), Academic Press, London, 1981.

[Li, 1987], Li, H., Maresca, M., Polymorphic Torns: A New Architecture for Vision Computation, Proc. 1987 Workshop on Computer Architecture for Pattern Analysis and Machine Intelligence, Seattle, WA, October 5-7, 1987, pp. 176-183.

[Kumar, 1987] Kuma, V.K.P.., Reisis, D., "Parallel Image Processing on Enhanced Arrays," Proc. International Conference on Parallel Processing, St. Charles, IL, August 17-21, 1987, pp. 909-916.

[Levitan, 1984] Levitan, S. P., *Parallel Algorithms and Architectures: A Programmers Perspective*, Ph.D. Dissertation, Computer and Information Science Department, also, COINS Technical Report 84-11, University of Massachusetts at Amherst, May 1984.

[Nii, 1986] Nii, H.P., The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures, AI Magazine, Vol 7, Number 2, pp.38-53.

[Random, 1987] The Random House Unabridged Dictionary of the English Language, L. Urdang, S. Flexner, Eds., Random House, N.Y., 1987

[Stout, 1986] Q.F. Stout, "Meshes with Multiple Buses", Proceedings of the 27th IEEE Symposium on the Foundation of Computer Science, 1986, pp. 264-273.

[Weems, 1984a] Weems, C. C., *Image Processing on a Content Addressable Array Parallel Processor*, Ph.D. Dissertation Computer and Information Science Department, also, COINS Technical Report 84-14, University of Massachusetts at Amherst, September 1984.(a)

[Weems, 1984b] Weems, C. C., Levitan, S. P., Foster, C. C., Riseman, E. M., Lawton, D. T., and Hanson, A. R., Development and Construction of a Content Addressable Array Parallel Processor (CAAPP) for Knowledge-Based Image Interpretation, *Proc. Workshop on Algorithm-Guided Parallel Architectures for Automatic Target Recognition*, Leesburg, VA, July 16-18, 1984, pp. 329-359.(b)

[Weems, 1985] Weems, C. C., The Content Addressable Array Parallel Processor: Architectural Evaluation and Enhancement, *Proc. IEEE International Conference on Computer Design: VLSI in Computers*, Port Chester, New York, October 7-10, 1985. pp. 500-503.

# ALGORITHMS AND ARCHITECTURES FOR SMART SENSING

**Peter J. Burt**

David Sarnoff Research Center, Subsidiary of SRI International
Princeton, NJ 08543-5300

## ABSTRACT

Human vision is a distinctly dynamic process in which the eyes move through a sequence of fixations to selectively gather scene information for the task at hand. At DSRC we are developing a set of similar strategies for computer vision. Included in these *smart sensing* techniques are mechanisms analogous to peripheral alerting and foveal attention in humans. Perhaps most importantly, smart sensing requires a close interaction between low and high level vision mechanisms, as image understanding processes guide information gathering processes even as initial image data is being interpreted. Smart sensing techniques can have a profound effect on the efficiency of a vision system, increasing its speed often by several orders of magnitude.

The multiresolution, pyramid structure provides a computational framework for smart sensing. We are developing computer architectures for hierarchical image analysis that support smart sensing, and we have built hardware to perform multiresolution image analysis at video rates.

## I. INTRODUCTION

This paper has two objectives. First, we provide an overview of major research projects at David Sarnoff Research Center (DSRC) that relate to image understanding. Second, we describe in some detail a project to develop computer architectures for multiresolution image analysis, and to build machines for a variety of vision applications.

Our interest is in developing a fundamental understanding of vision, in terms of representations, algorithms, reasoning structures, and architectures. Our current architecture project is an outgrowth of a number of diverse research programs, in human vision, signal processing and image understanding. To a large extent these projects have been tied together by an interest in a certain style of computation, that is hierarchical, 'flow through', and controlled by strategies of 'smart sensing'. This interest reflects a conviction that requirements for efficient computation place fundamental constraints on the form taken by a vision system. It is the need to make the very most of limited resources, that has dictated some of the most striking features of natural vision, including the foveal organization of the eye, and alerting and attention mechanisms. Similarly, the basic requirements for efficiency have led us to architectures based on hierarchical processing and smart sensing.

The overview of DSRC vision related programs is given in Section II of this paper. Discussions of smart sensing and multiresolution processing are given in the next two sections. These motivate the architectures and machines described in the remaining sections.

## II. IU RESEARCH AT DSRC

The following paragraphs provide a brief overview of major image understanding (IU) and related programs at the David Sarnoff Research Center.

• **Smart Sensing.** We define smart sensing as the selective gathering and analysis of scene information as required to perform specific vision tasks. It is implemented within a vision system as a set of techniques and strategies that yield speed and efficiency in analysis. These include, for example, alerting, foveation and attention mechanisms. Smart sensing is essential for practical vision systems that must work in real time. Our objective is to develop a unified set of techniques that may then be applied to a wide range of tasks, including industrial inspection, surveillance, real time motion analysis, ALV navigation and general outdoor vision. [12][13]

• **Architectures for Multiresolution Vision.** A key element of smart sensing is the ability to represent and process image data at multiple resolutions. Typically this yields improvements in speed or reductions in hardware by three or more orders of magnitude. We have developed formal models for several aspects of multiresolution processing, and are formulating vision architectures based on these models. Of particular interest are architectures that are modular and *flow-through*. These can be configured as very small, special purpose designs, or large, general purpose, machines for real time vision tasks. *Lattice pipeline* and *circulating pipeline* architectures will be reviewed in this paper. [20][21]

• **Pyramid Vision Machine.** Based on the general concepts developed in the architecture program, we are building several machines for a range of specific applications. The first of these is called the Pyramid Vision Machine, or PVM, and is intended primarily for industrial applications. The machine is relatively small but can achieve remarkable speed through the application of smart sensing strategies. A prototype PVM was completed in 1985, and a second generation machine is now under development. [5][38]

• **Real Time Motion Analysis.** A second machine based on hierarchical, flow-through architectures is being devel-

oped for continuous, real time analysis of motion in a video sequence. This machine will be built with the same processing modules as the PVM. Initially it will be applied to motion directed processing of television signals, but applications to computer vision are planned, including object tracking and ALV navigation. [7][22]

• **Surveillance.** A third machine, also based on PVM components, is under development for 'smart surveillance' applications. This is a very small device that could ultimately be part of the camera itself. Again using smart sensing strategies it will monitor motion within a camera's field of view, discriminate between possible target motion and other background actively, and locate and track selected targets. [4][17]

• **Princeton Engine.** A very high speed parallel processing machine, known as the Princeton Engine, is under development at DSRC and will be completed in mid 1988. This is a coarse-grained SIMD machine designed to perform relatively complex iconic processing on video signals in real time. High speed data paths move images through the machine. In its full 2048 processor configuration the machine will operate at 30,000 MIPS. [25]

• **Object Recognition.** A major objective of our research program is the development of systems that can locate and recognize objects of interest within complex scenes at near real time. Again, the techniques of smart sensing are critical for speed. Emphasis is given to the design of model-based reasoning structures that can rapidly assimilate new scene information in order to direct the ongoing information gathering processes. As an initial implementation we are developing hierarchical *pattern tree* object models that combine a compact object description with object specific recognition strategies. [13]

• **Human Vision.** DSRC has had a significant basic research program in human vision for many years. In the 50's and 60's Rose and Shade were first to characterize human perception in terms of signal-to-noise limitations and modulation transfer functions. These powerful notions led others to develop the now familiar multi-channel models of human vision. Members of our staff have contributed to these later endeavors, and we continue to extend vision models to describe such basic functions as the perception of motion, depth, and texture. Our program includes applied as well as basic research, and models of human vision have been used to measure image quality for video displays and to design industrial computer vision systems for the detection of cosmetic flaws. [1][2][6][8][9][18][19][23][24][29][35][36][39][40]

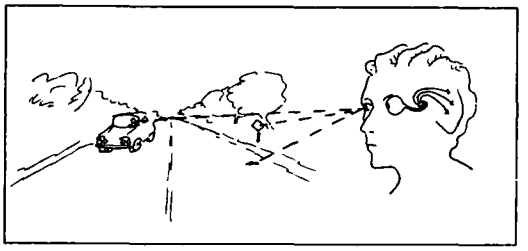
• **Neural Networks.** We are developing neural net approaches to various vision related computations. One objective of our current work is 'data fusion', the combining of information from different sensory modalities, including tactile and auditory senses as well as vision. Our approach includes the development of biological models for brain structures that perform these functions as well as computer architectures for the same functions. The objective of the program ultimately is the development of neural techniques and architectures for object recognition and robotic object manipulation. [33][34][37]

• **Image Representations.** The representation of image information is key to the success of analysis. A good representation will not only capture salient information in a easily accessible form, but will support rapid search and analysis algorithms. DSRC has been active for a number of years in the development of multiresolution, pyramid based, image representations. These have proven useful not only for computer vision, but in signal processing in general. Major applications outside vision include image compression, enhancement, and graphics. [15][16]

• **Joint University Programs.** DSRC has active or developing vision related programs with several universities. These include the University of Pennsylvania (algorithms), NYU (representations), Princeton (robotics, neural biology), and Rutgers (machine vision).

## III. SMART SENSING

While vision research at DSRC spans many disciplines, from algorithms to hardware, much of this work is developed within a common conceptual framework. Our objective is the development of highly efficient vision systems that can perform significant vision tasks in near real time. To a large extent computational strategies for this performance are captured in the notion of smart sensing. [12][13]



Smart sensing may be defined as the selective, task oriented gathering of information from the visual world. It is a distinctly active process in which the viewer, be it man or machine, probes and explores the visual environment to extract information for the task at hand. Such selective processing is essential for any vision system to function in a complex environment. The visual world contains vast amounts of information, often far more than can be analyzed even by today's supercomputers. However, only a minute fraction of this information is required to perform most vision tasks. When just this salient information can be efficiently located and extracted then even a modest computer can perform sophisticated tasks.

Many aspects of smart sensing are already familiar to us from our own visual experience. Imagine, for example, the driver of a car traveling on a country road. His visual task is to locate and follow the road, observe oncoming traffic, read appropriate road signs, and avoid any objects in the road. To perform this task he does not examine the world in uniform detail. Rather he moves his eyes to fixate in turn certain critical points in the visual field, the road, an oncoming car, a sign. Two or three such fixations per second suffice to drive the car, yet he 'sees' only a minute fraction of the world before his eyes.

The eye is capable of gathering detailed scene information only in the small region centered on its fovea. Its resolving power falls rapidly outside this region, towards peripheral vision. Thus while the driver fixates the on road he hardly notices individual leaves, or even trees by the side of the road, or objects in the fields beyond. What is remarkable is the skill with which his visual system locates and extracts just that information required to drive the car, while it ignores the vast flow of information not relevant to this task.

The same behavior is evident when humans perform an industrial vision task. Objects to be inspected, for example, may be large and intricate, but a human inspector quickly locates just those points that need close examination. As machines are designed to perform such inspection tasks it will be imperative that they also implement smart sensing strategies. Even the relatively constrained industrial environment is too complex for exhaustive visual analysis.

What are the basic elements of smart sensing? In the case of human vision three such elements are apparent in the driver example. First is the marked concentration of resolving power within the fovea of the eye. This serves at once as a probe by which the visual system can explore the world, and as a mechanism by which it can limit the amount of detailed information it must process at any moment in time. Second is the broad field of view provided by peripheral vision. This serves alerting and guidance functions. While sensing the world at much reduced resolution peripheral vision can detect unexpected objects or events, and can direct the fovea for close examination.

A third element of smart sensing evident in the driver's behavior is the high level cognitive control of eye movements. As a viewer performs a task such as driving, he understands its requirements and moves his eyes to gather the specific information needed for the task. This aspect of natural vision is most remarkable, and most unlike present day computer vision systems. It reveals a tight coupling and dynamic interplay between high and low levels of visual analysis. Even as the low level system gathers pieces of scene information through the sequence of foveations, the high level system is attempting to interpret the world based on this partial data. In effect the high level system continually forms and reforms hypothesis about objects in the scene, and directs the eyes to gather additional information to confirm or reject these hypotheses.

Ultimately smart sensing must be understood in terms of computational efficiency. It is based on analysis strategies for rapid identification of the salient information within a scene, and on data structures and fast algorithms for effective extraction and representation of this information. To a significant degree, the form taken by a vision system is dictated by these structures and strategies that serve fast analysis.

Analogous structures and strategies support smart sensing in computer vision. Five basic elements of smart sensing are as follows:

- ***Match to Resolution and Scale***

The resolution required to perform a given vision task is not fixed, but may vary from moment to moment and from region to region over a scene. In some cases only coarse pattern structure needs to be measured over a relatively broad area of the visual field; in other cases details are required in small patches of the field. In computer vision, it is important to match the scale and resolution of analysis operations to that of salient information within the scene. Not to do so can mean the vision system becomes mired in irrelevant detail, 'missing the forest for all the trees'.

As a simple but telling demonstration of the importance of scale and resolution, consider the problem of locating a target pattern, **T**, within an image **I**. If the target measures $M$ by $M$ samples, and the image measures $N$ by $N$, then the cost of correlation search is order $M^2N^2$. Next suppose we wish to search for the same target but increased in scale by a factor $s$. Two approaches may be considered. In the first a larger copy of the target pattern is constructed, and the search is repeated as before. The cost is now order $s^2M^2N^2$. In the second approach, the image is decreased in scale by $s$ and correlation is repeated. In this case the cost is decreased to order $M^2N^2/s^2$. Provided the target is represented in sufficient detail to permit identification, the two methods will yield the same results. The cost of computations in the first approach, however, exceeds that of the second by a factor $s^4$. This can indeed be large for scale factors of 10 or more that are common in vision tasks! It can be shown that when searching for targets over a range of scales and orientations the cost is proportional to the $6^{th}$ power of the target dimensions (in samples). [refs. Proc.] The importance of performing analysis at minimum size and resolution is clear.

- **Foveation.**

It is the nature of the physical world that objects of interest are localized in space and events are localized in time. In computer vision it is expedient, therefore, to focus analysis just on those regions of the visual field that carry salient information, and to move the analysis region dynamically as events unfold. This dynamic allocation of computer resources to regions of interest is analogous to foveation in humans. Likewise, the additional analysis steps applied within a region of interest is analogous to attention in humans. In computer vision, however, many such analysis regions may exist at once, and they may change in size and resolution as required.

Mechanisms of foveation and attention can yield very significant improvements in the efficiency of a vision system. They require, however, that alerting and guidance mechanisms be implemented over the full field of view to detect events of potential interest and direct foveal analysis. They also require high level, model based, guidance as outlined below.

• **Fine-to-Coarse Generation of Feature Sets.**

Selective visual analysis is generally not applied to raw image data, but to data that has been preprocessed in various ways to enhance features important to interpretation, to compute local measures, such as texture or motion characteristics, and to adjust resolution appropriately for analysis. Highly efficient fine-to-coarse hierarchical algorithms can generate many such feature images rapidly and at modest computational cost.

• **Coarse-to-Fine Search.**

It is often possible to achieve efficiency by performing a given analysis task first on the image represented at low resolution. Results of this initial analysis are then refined in a sequence of steps while moving to progressively higher image resolution. Cost is minimized at low resolution because data is represented at low sample density, and minimized at successively higher resolutions because results at one resolution guide processing at the next. Such coarse-to-fine strategies are commonly used in motion and stereo analysis, and are important in guiding 'foveal' vision in object recognition.

• **Model Based Control.**

As is evident in the example of the automobile driver, humans have a remarkable ability to direct their eyes to those regions of the visual field where important information is likely to exist. The same should be true with smart sensing in computer vision. The selective gathering of scene information must be guided to a large degree by an understanding of the visual scene and objects that may be within it. There should be, therefore, a tight coupling between the interpretation processes of a vision system and the information gathering processes. Perhaps the most challenging task in implementing this aspect of smart sensing is the development of object models that permit very rapid assimilation of new information and reasoning structures that support rapid decision processes to direct further steps in the selective gathering of scene information.

## IV. A COMPUTATIONAL FRAMEWORK

The computational elements of smart sensing will be illustrated in the next section with a number of examples. Here we introduce notation for computations based on the image pyramid since this will provide a computational framework for smart sensing.

The pyramid is, first of all, an image transformation that is compact, and complete, and that in its various forms isolates and enhances image features, such as edges, that are important in analysis. As a multiresolution representation, the pyramid provides a data structure in which analysis operations can be matched to the scale and resolution of salient image patterns. Furthermore, because the structure is hierarchical, it provides the computational framework for fast fine-to-coarse algorithms that generate local image measures, or features sets, and fast coarse-to-fine search procedures for locating features or objects of interest. All of these techniques are essential for smart sensing, and all may be combined elegantly within the pyramid structure. [10][11]

### Definitions

A pyramid is a sequence of images of decreasing resolution obtained by repeatedly convolving a filter **w** with an initial image. The unique feature of this *hierarchical convolution* procedure is that **w**, called the *generating kernel*, is 'expanded' with each iteration. This means the distance between filter taps is increased by a constant factor, typically 2, with each filter stage. In the basic Gaussian pyramid, **w** acts as a low pass filter. Successive doubling of **w** means that the band limit is reduced in octave steps from image to image in the pyramid sequence. As the band limit is reduced, sample density may also be decreased by the same factor of two in each dimension. This subsampling is optional, however. We say that the image sequence is a 'full density' pyramid if it is not subsampled, or a 'standard density' pyramid if it is.

A Gaussian pyramid may be viewed as a discretely sampled 'resolution space'. In the case of a standard density pyramid, subsampled levels may be further compressed into smaller arrays. The scale of images and image patterns is then reduced by two with successive level. This subsampled and compressed pyramid represents a discretely sampled 'scale space'.

While the generating kernel, **w**, is typically small, successive doubling in the hierarchical convolution procedure means its effective size grows exponentially. Let $\mathbf{w}_\ell$ be an *expanded kernel* in which taps are separated by a distance $2^{\ell-1}$:

$$\mathbf{w}_\ell(m,n) = \begin{cases} \mathbf{w}(\frac{m}{2^{\ell-1}}, \frac{n}{2^{\ell-1}}), & \text{for } m \text{ and } n \text{ multiples of } 2^{\ell-1} \\ 0, & \text{otherwise.} \end{cases}$$

Then the levels of a Gaussian pyramid, $\mathbf{A}_0$, $\mathbf{A}_1$, ..., $\mathbf{A}_\ell$, ..., formed form an original image, **A**, are defined recursively: $\mathbf{A}_0 \equiv \mathbf{A}$, the original image, and for $\ell > 0$

$$\mathbf{A}_\ell = \mathbf{w}_\ell * \mathbf{A}_{\ell-1}.$$

The $\ell^{th}$ pyramid level is obtained as a cascade of $\ell$ convolutions with the original image:

$$\mathbf{A}_\ell = \mathbf{w}_\ell * \mathbf{w}_{\ell-1} * \ldots * \mathbf{w}_1 * \mathbf{A}$$

Alternatively we may express this as a single convolution with a *hierarchical kernel*, $\mathbf{W}_\ell$:

$$\mathbf{A}_\ell = \mathbf{W}_\ell * \mathbf{A}$$

where

$$\mathbf{W}_\ell = \mathbf{w}_\ell * \mathbf{w}_{\ell-1} * \ldots * \mathbf{w}_1.$$

Although the hierarchical kernel is never explicitly computed in pyramid image processing, it is convenient for describing the net result of a hierarchical convolution. Hierarchical kernels double in size with each iteration, but for a given generating kernels, their shape is fixed. For example, a commonly used choice of generating kernel yields Gaussian-like hierarchical kernels of many scales.

In computer vision, Gaussian pyramid construction is often performed as a first step in constructing other types of

image pyramids. Of particular importance is the Laplacian, or band pass, pyramid. If $\mathbf{G}_\ell$ is the $\ell^{th}$ level of a Gaussian pyramid, then the corresponding Laplacian level, $\mathbf{L}_\ell$, is given by:

$$\mathbf{L}_\ell = (1 - \mathbf{w}_\ell) * \mathbf{G}_\ell.$$

Levels of the Laplacian pyramid represent differences between successive levels of the Gaussian pyramid.

Gaussian pyramid construction is also frequently used to integrate local image properties within Gaussian-like windows of many sizes. This *hierarchical integration* process will be illustrated in the next section.

It is useful to adopt a shorthand notation for Gaussian and Laplacian pyramid processes since these will be the basis for examples given below. Let $G_\ell[.]$ and $L_\ell[.]$ be operators that construct the $\ell^{th}$ Gaussian and Laplacian pyramid levels from any image specified in the brackets. For example, $\mathbf{A}_\ell = G_\ell[\mathbf{A}]$. Similarly $\mathbf{E}_{m,n} = G_n[(L_m[\mathbf{A}])^2]$ indicates that an 'energy image', $\mathbf{E}_{m,n}$, is constructed in three steps: first Laplacian pyramid level $m$ is generated for an original image $\mathbf{A}$.; second, the samples of this band pass image are squared; and finally, local energy measures are formed by integrating the squared Laplacian values within windows $\mathbf{W}_n$ by constructing a Gaussian pyramid to level $n$.

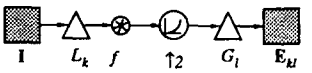
No distinction is made in this notation between pyramids represented at standard and full density. Because images are band limited, and standard sample density approximates the Nyquist sample rate, both pyramid forms accurately represent the same underlying signal. In vision applications, however, efficiency is often critical, and subsampling is almost always used. In practice, intermediate results in multiresolution computations are often represented at twice the standard sample rate, in 'double density' pyramids. This is because certain iconic operations, such as squaring or image-image multiplications, double the band width of the resulting image.

### Image Flow Diagrams

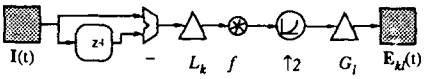
Iconic analysis procedures for early vision can often be represented concisely by *image flow diagrams* such as those shown in Figure 2. Each operation is represented by a symbol, Figure 2a, while the sequence of operations required for a processing algorithm is indicated by arrows between symbols. One may imagine a processing network in which images move between physical computing modules as they undergo processing steps. Indeed this view will be taken quite literally in defining architectures for hierarchical iconic analysis.

Images entering and exiting the network are shown as shaded rectangles in the diagrams. Often an image is decomposed into sets of images through pyramid processing, then subsequent operators are performed on all images in the sets. Open arrows are used to indicate paths on which such image sets move together. Multiple, overlapping symbols may be used to indicate operations performed on each image in these sets.

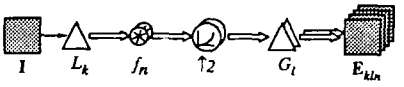
2a: Symbols



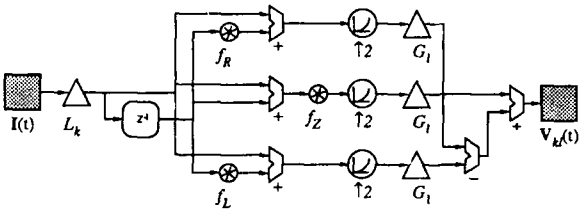
2b: Integrated Measure



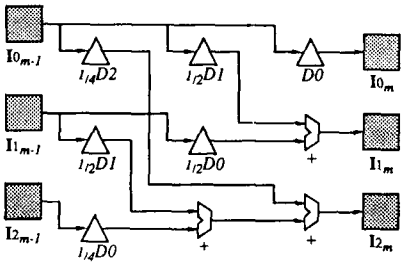
2c: Change Energy Measure



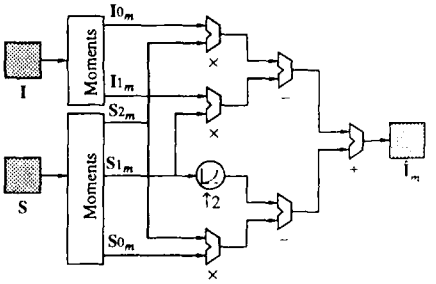
2d: Feature Sets



2e: Motion Energy



2f: Moments



2g: L... or Fit Filtering

Figure 2: Image Flow Diagrams

Image flow diagrams without loops define a class of operations that are particularly important when vision tasks must be carried out continuously, in real time. Images may then enter the network, flow through its processing stages, and exit without interruption in processing. We refer to this as *flow through* analysis.

## V. EXAMPLES

In this section we present a number of examples to illustrate the basic computational elements of smart sensing. These examples also help motivate the architectures presented in the next section.

### A. Multiresolution Representations

It is often useful to view a pyramid alternately as a computational structure and as an image representation. As a representation the pyramid and related hierarchical structures have attracted considerable interest in recent years for a wide range of applications. The Laplacian pyramid, for example, represents an image as a set of sampled, band pass images, or equivalently, as an array of Gaussian-like basis functions of many sizes. Gradient and oriented Laplacian pyramid decompose images into primitives indicative of edge features. The Laplacian pyramid has found important applications in image compression and graphics.[15][16] Gradient pyramid have been applied quite successfully to image enhancement.[31] QMF pyramids, based on quadrature mirror filters, have recently proven particularly effective for image compression. [3][32]

Related hierarchical representations in term of self similar basis functions have attracted considerable interest in scientific fields far removed from image analysis. For example, 'wavelet' representations are proving to be powerful tools in the study of quantum physics and geology. [26][27][28]

### B. Image Detail at Reduced Resolution

It is often assumed that multiresolution techniques are not applicable to many of the most challenging vision task, such as the analysis of satellite images, because critical features are very small and are lost if the image resolution is even slightly reduced. In fact such fine pattern details can often be represented at low resolution through conversion to *integrated measures*.[11][13] These record the presence of small features within a region of the image, but represent their position with reduced precision. The use of such measures can be particularly important in the analysis of very large format images, or in recognizing objects under variations in position, scale and orientation.

Four distinct steps are typically required in generating an integrated measures, Figure 2b. First, the source image is decomposed into band pass components through Laplacian pyramid construction. Second, the component best matched in scale and resolution to the features of interest is selected for further processing. This image is convolved with a filter, $f$, that enhances the feature of interest. Third, the resulting 'feature image' is typically band pass, with both positive and negative values. The image is converted to strictly positive values through a rectification operation, such as squaring its sample values. Finally, local integration is used to group, or sum, occurrences of the pattern element within small neighborhoods. Integration is achieved through the construction of a Gaussian pyramid. For integration within window $\mathbf{W}_\ell$ the $\ell^{th}$ Gaussian level is selected as the final integrated measure.

An integrated measure image is analogous to an array of complex cells in the human visual system. An individual complex cell becomes active if the particular pattern for which it is selective, such as an oriented line element, occurs anywhere within an extended receptive field. Cell activity represents the fact that the pattern has occurred in the image but only specifies its location roughly, as being somewhere within the cell's receptive field. In the same way, a positive value of an integrated measure indicates that a selected feature occurs somewhere within its integration window.

### C. Alerting and Foveation

Perhaps the most familiar aspect of smart sensing in natural vision is the foveal organization of the eye itself. Through the concentration of resolving power within the fovea, the system can selectively gather scene information through a sequence of fixations. At the same time peripheral vision monitors a wide field of view at low resolution, providing alerting and guidance for foveal examination. A similar allocation of resolution and analysis is important in computer vision to achieve an efficient use of computing resources.

Integrated measures provide a basis for alerting and guidance. This can be illustrated with an application developed at DSRC to 'smart Surveillance'.[4][17] Suppose that an automated surveillance camera is required that can monitor an outdoor scene, and detect when certain critical events occur while ignoring uninteresting background activity. Suppose further that the camera-analysis system must be low cost, and hence that it can have only limited computing power.

We have developed a prototype smart surveillance camera consisting of specialized pyramid hardware (the PVM, described below) and a standard microprocessor. The pyramid hardware is used to compute integrated measures over the full field of view at video rates. These 'change energy' measures serve as the basis of alerting and guidance, directing simple analysis programs in the microprocessor to examine small regions of the filed of view that contain suggestive activity.

The basic computation used in the change energy generation is shown in Figure 2c. A difference image is first formed between successive frames of a video sequence. This is nonzero only where changes are taking place in the scene. The difference image is decomposed into a set of band pass components, filtered, squared, and integrated, as in the standard procedure for integrated measures. The resolution band, $k$, and filter, $f$, are chosen to be selective for events of interest. All levels of the integrated pyramid are made available to the microprocessor for analysis. Because the microprocessor has limited computing power it has time to examine only

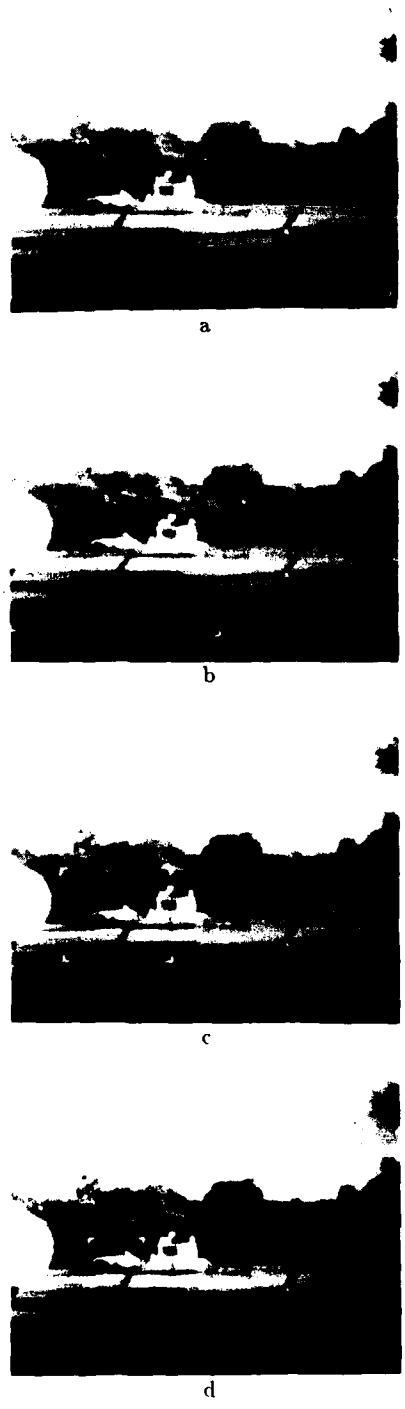



Figure 3. Homing process over four frames. Corner marks indicate the image region examined each frame time. The region becomes smaller, but represents higher resolution, as analysis moves between pyramid levels.

a small 16 by 16 region of the change energy pyramid each frame time. In the normal mode of operation, the microprocessor examines only the 16 by 16 low resolution pyramid level. Any change in the scene with the appropriate characteristics can be detected by the microprocessor at this level, even changes in single pixels, if these are deemed interesting. If a possible event is detected the system 'foveates' the appropriate region of the visual field through a homing process. After detection within a 16 by 16 window at level $\ell$ in one frame time, the system shifts in the next frame time to examine a 16 by 16 analysis region at level $\ell - 1$. This analysis region is centered on the location of the detected activity at level $\ell$, and represents a smaller area of the scene at higher resolution. The homing process is illustrated in Figure 3.

In this example a single integrated measure was used to guide 'foveal' analysis. More generally a vision system will base alerting and guidance on a set of different integrated measures. Through appropriate combinations of these measures the system can be made selective for a very limited class of events or a broad range of events as required.

### D. Energy Measures and Feature Sets

Hierarchical computations are also an important source of local image measures used in higher level image analysis. Random textures, for example, are often best characterized at two scales, a finer scale corresponding to pattern elements that make up the texture, and a larger scale corresponding to the regions over which the statistical distribution of the elements can be measured. Measures of local displacement in stereo and motion analysis are also most economically computed within a multiresolution structure. Larger displacements are represented economically at lower resolutions while small displacements are represented at higher resolution.

Figure 2d shows a general framework for generating texture measures. This is like the basic integrated measure computation, Figure 2b, except that each band pass pyramid level is processed with a set of filters $\{f_n\}$, and the resulting energy images are formed with integration windows of many sizes. The result is sets of energy images, $\{E_{k\ell m}\}$, where $k$ indicates feature scale, $\ell$ indicates the integration window size, and $n$ the feature type.

The importance of this computation lies in part in the fact that the computations are highly efficient and regular. Efficiency is the result of its hierarchical organization: complex measures within large windows are computed as sums of simpler measures within smaller windows. Intermediate results are used repeatedly in the formation of later results. In this way, it is often possible to generate multiple features at multiple scales and within windows of many sizes at a cost that is only slightly greater than that of computing just one feature image at a selected scale and window size. [11][30]

*Julesz Textures.* Texture energy measures can also be used to model texture perception by humans. Recently Bergen and Adelson have shown that a simple measure based

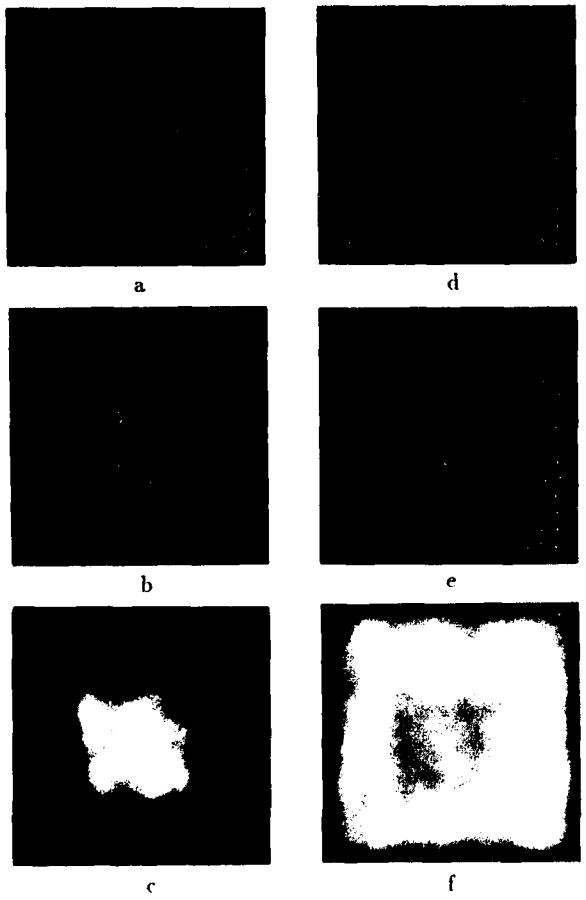



Figure 4. Local energy measures used to discriminate Julesz textures.

just on the Laplacian pyramid can predict human discrimination of certain Julesz textures.[6] For example, regions of 'L's and 'T's in Figure 4a are readily discriminated by humans although these regions have been carefully balanced to have identical (global) first and second order statistics. This means their power spectra are identical. Locally, however, the texture elements differ in spectral characteristics. This is shown in Figures 4b and c. Figure 4b is Laplacian level 3 squared, while Figure 4c shows the same level integrated within a Gaussian window: $\mathbf{E}_{3,5} = G_5[(L_3[\mathbf{I}])^2]$. The two texture regions are clearly distinguished by differences in local energy within this band.

The above observations become more interesting in light of a second demonstration shown in Figure 4d to 4f. The texture in Figure 4d is a modified version of the texture in 4a in which the 'X's are made slightly larger and the 'L's slightly smaller. Now first and second order statistics no longer match, yet the discrimination is significantly more difficult for humans. Figure 4e and 4f show that $(L_3[\mathbf{I}])^2$ and $\mathbf{E}_{3,5}$ also fail to discriminate these textures. The implication is that local energy measures, such as $\mathbf{E}_{3,5}$ are used by humans in texture discrimination rather than direct measures of average intensity or global second order statistics.

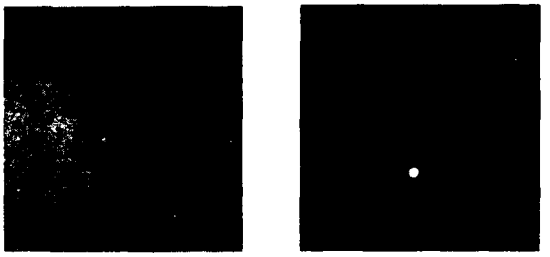



Figure 5. Surface flaw inspection. (a) Original with watermark. (b) Flaw revieled by local energy measure.

*Motion Energy.* Energy measures can be defined for motion and stereo in much the same form as for texture. [22] Figure 2c shows an image flow description of a method for computing motion energy and local velocity based on the work of Bergen and Adelson.[7]

*Blemish Detection.* As a final example, Laplacian energy measures can be used in industrial inspection to detected blemishes based on a model of their perceptibility by humans. Figure 5a shows a portion of a CRT screen that must be inspected for flaws in the deposition of a phosphor coating in the manufacturing of TV sets. A diffuse flaw of a type known as a 'water mark' is present. While the flaw is relatively easy for humans to locate, it can be quite difficult for computers. Flaws of this type have low contrast, here only 4 gray levels, or 2% relative to the surrounding regions. And other features of the CRT image tend to mask the flaw, including a fine grain texture and a gradual but significant gradation in brightness over the CRT screen. The presence of these background characteristics precludes the use of simple thresholds, or other pixel resolution operations.

However the flaw can be detected quite readily on the basis of local spectral energy. This is apparent in Figure 5b which shows texture energy $\mathbf{E}_{2,3}$ based on Laplacian level 2 integrated to Gaussian level 4.

Blemish detection is a particularly challenging problem in industrial inspection because flaws are cosmetic in nature, and are significant only to the degree they can be seen by humans. The compu'er vision system must therefore model human vision. The simple energy based system described above provides just such a representation. The human visual system has been modeled as a set of frequency tuned channels that differ systematically in their sensitivity.[40] These channels can, in turn, be modeled by levels of the Laplacian pyramid with energy measures weighted appropriately to predict human discrimination.[23]

### E. Processing Incomplete Image Data

In all examples described here the computations used to derive one image from another are 'local' and 'homogeneous'. That is, each output image sample is based only on input samples within a neighborhood centered on the corresponding position in the input image, and the same basic computation is repeated for all output samples. It is possible to construct processing that is locally adaptive

by appropriately combining such homogeneous operations. It is even possible to perform regular operations on irregularly sampled image data. This is necessary, for example, when applying filter operations to image segments of irregular shapes, or to motion or stereo displacement estimates that are known only at scattered points of an image. A general procedure for both filtering and local integration involves fitting polynomials to the available image data, then performing the desired operations to the extrapolated data (Burt, 1988). A highly efficient procedure for obtaining best fit polynomials within local neighborhoods of each output sample begins with the computation of local image moments. Such moments are useful in other aspects of image analysis as measures of local pattern properties.

A normalized order $p$ moment within window $\mathbf{W}_\ell$ centered at point $(i)$ is defined as

$$\mathbf{I}p_\ell(i) = \frac{1}{2^p} \sum_m \mathbf{W}_\ell(m)\mathbf{I}(i+m)m^p.$$

(The definition is given in one dimension to simplify notation.) Note that a moment is defined with the window centered at each sample point. It can be shown that the $p^{th}$ moment in widow $\mathbf{W}_\ell$ can be computed as a simple combination of $p$ and lower order moments in window $\mathbf{W}_{\ell-1}$ [ref.]:

$$\mathbf{I}p_k = \frac{1}{2^p} \sum_{q=0}^{p} \binom{p}{q} \mathbf{w}r_k * \mathbf{I}q_{k-1}.$$

Here $\mathbf{w}r$ is a *moment generating kernel* which combines a moment arm with the standard Gaussian generating kernel:

$$\mathbf{w}r(m) = \mathbf{w}(m)m^r.$$

In the above equation $r = p - q$. This is a fast, robust, computation when performed with a standard density pyramid.

The computation of zero, first and second order moment images is shown as an image flow diagram in Figure 2d. Here the pyramid operations labeled $\mathbf{D}p$ are the same as in standard Gaussian pyramid construction except that moment generating kernels are used.

A polynomial fit within window $\mathbf{W}_\ell$ centered at image point $i$ can be obtained directly from the corresponding moments. Let $\mathbf{S}$ be a support image with value '1' everywhere samples of source image $\mathbf{I}$ are known. Let $\hat{\mathbf{I}}_m$ be the image obtained through polynomial fit filtering. Then

$$\hat{\mathbf{I}}_m = \frac{\mathbf{I}0_m\mathbf{S}2_m - \mathbf{I}1_m\mathbf{S}1_m}{\mathbf{S}0_m\mathbf{S}2_m - \mathbf{S}1_m\mathbf{S}1_m}.$$

The image flow diagram in Figure 2g shows that linear fit filtering begins with parallel computation of moments for the source and support images, followed by several stages of image-image arithmetic operations.

### F. Structured Search and Fast Object Recognition

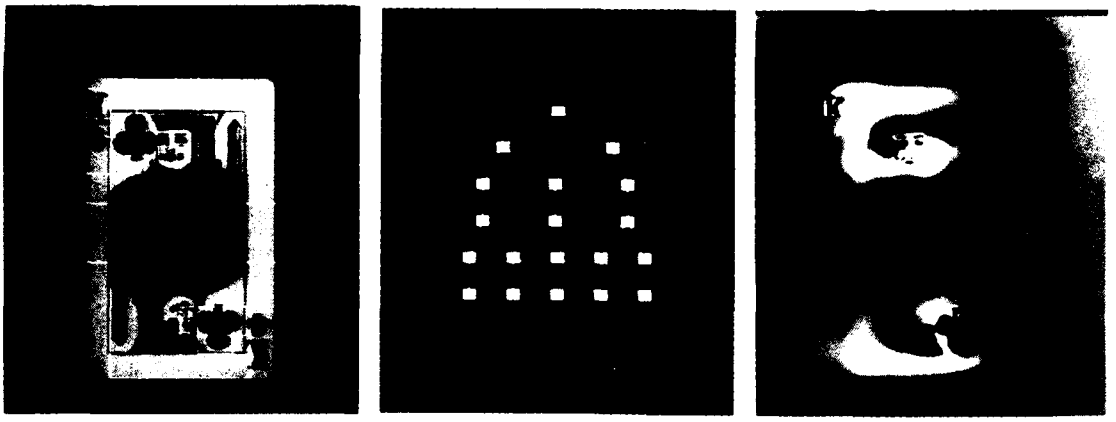
A tightly coupled interplay between high and low level analysis is key in smart sensing. If a system is to be effective in gathering information from a scene, it must anticipate where that information is likely to occur. Such anticipation must in turn be based on internal models of the objects of interest within a scene. Even as these models direct the searching process they must be refined and updated to reflect newly gathered scene information.

To implement model directed sensing within a computer vision system it is essential that model and reasoning structures be carefully designed to support very rapid decision and update processes. Information gathered from one 'foveation' must be assimilated and lead to quick decisions about where to direct the next 'foveation', typically within a fraction of a second. Needless to say, such dynamic interplay between reasoning and sensing is not characteristic of present day computer vision machines. Its implementation will require innovations particularly in the design of object models and reasoning structures.

At DSRC we have begun the development of model based smart sensing with hierarchical object models called *pattern trees*. While still in a relatively early stage of development, this structure is already suited for certain industrial vision tasks. [13] In its simplest form the pattern tree represents an object in terms of a set of characteristic features over a range of scales. In general, large features are represented at low resolution, while small features are represented at high resolution. Typically, the root of the tree corresponds to the overall pattern shape, while branches represent local features in progressively more detail. Recognition processes begin at the low resolution description. The pattern tree provides both a compact object description and the control structure for directing object recognition processes. The decomposition of the pattern into parts provides efficiency and flexibility. As was observed earlier, the cost of pattern search over positions, orientations, and scale, can grow as the $6^{th}$ power of the pattern size. The decomposition of large, complex patterns into smaller, more primitive patterns, can yield remarkable savings in computations. At the same time the decomposition directs analysis to the most significant features of the object, and provides the framework for handling object occlusion.

The pattern tree supports the desired interaction between sensing and reasoning processing. Initial feature information (e.g., from alerting mechanisms) is used to select candidate models for objects that may be in a scene. Models are reinforced and refined as sensing process gather scene information, while the branching structure of the pattern tree serves to guide the selective search.

An example pattern tree description of a playing card is shown in Figure 6. Boxes drawn over the pattern, Figure 6a, show regions represented as features in the pattern tree. In this case all pattern components are represented directly as sample arrays. Here all arrays measure 8 by 8 samples, so larger boxes correspond to lower resolution subpatterns in the tree. The tree itself is shown in Figure 6b. Figure 6c shows the pattern information actually included in this tree, obtained by summing its subpatterns.



a b c

Figure 6. Pattern tree representation of a playing card.

## VI. ARCHITECTURES

We now consider architectures for vision systems designed to perform multiresolution images analysis guided by the principles of smart sensing. A number of the structural and functional requirements of such an architecture are evident form the forgoing discussions. These include some functions for which existing architectures are well suited. For example, all iconic operations discussed thus far have been 'local', 'homogeneous' and 'registered'. The first two of these characteristics are familiar, and have been defined. To say that an operation is 'in register' means the output image has the same spatial coordinate as the input image. We will refer to this general class of computations as 'LHR' operations. Such operations are well suited for parallel computation on SIMD mesh architectures or on pipelined architectures.

Other structures and functions are not well suited for current architectures, particularly SIMD mesh designs. For example, smart sensing techniques described here rely heavily on the ability to process images at multiple resolutions and sample densities, and to restrict processing to regions of interest that are dynamically defined and redefined in the coarse of analysis. Mesh architectures are poorly matched to such operations. Subsampling and windows can be implemented through mask operations in which all processors are disabled except those corresponding to selected sample positions. But in doing so a system cannot take advantage of the computational efficiency that sampling and region of interest processing should afford. Reductions in efficiency and speed of several orders of magnitude would be typical. A further complication in the mesh implementation of hierarchical processing is that when processing low resolution data, processing elements that represent neighboring samples are far apart in the mesh. This necessitates either a transfer operation in which samples are repacked into neighboring processing elements, or long range communication of information between elements. Both alternatives are inefficient in a mesh machine.

A pipeline design provides an attractive choice for processing subsampled and windowed images. In this architecture, processing is performed as image data moves through processing elements. Changes in sample density and in (analysis) window size take place as data is in transit. The time required to complete an operation is directly proportional to the number of samples that must be processed. In this way thousands of operations can be performed on low resolution and windowed images in the time required to perform a single operation on the a image.

It might be argued that an SIMD mesh is inherently faster than a pipe because it has such large numbers of processors. If there are half a million processors performing a task in parallel, can it really matter if some are idle? Still, pipeline designs can also be made fast by running multiple pipes in parallel, and in practice the processors used in a pipe design are often significantly more powerful than SIMD processors. It seems, in fact, that when two machines are designed to perform LHR operations at the same overall speed, a pipeline design requires far less hardware than a mesh.[20]

Here we develop two architectures based on pipeline processing modules. The designs differ in the way image data is moved between these elements. The first design, called a *lattice pipeline*, offers minimal delay through a sequence of operations, and is appropriate for specialized systems that must process large amounts of data in real time. The second design, called a *circulating pipeline*, provides more efficient use of computational resources, and is suited for general purpose vision systems.

In both architectures we have favored modular design, so that large and small systems can be assembled from a common set of components, as required to meet the data rates for different vision tasks.

In practice, architectures that combine features of lattice and circulating designs are preferred. The PVM is one such hybrid design.

## Processing Modules

In order to define hierarchical iconic processing architectures in the most general terms, we begin by adopting a standard model for a processing element, as well as a standard symbolic notation, Figure 7a. It is assumed the module performs an iconic operation, generating an output image by applying an operation $f$ to an input image. In general, a module may operate on a number of input images, and generate a number of output images; however, we will consider modules with only one or two inputs and a single output.

A given module may be capable of performing a number of different operations, however it is assumed that when an operation is applied to an image, it is applied to the entire image. The operation is set by an external control line, as shown, and may be changed from image to image.

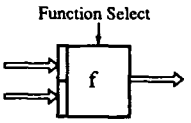



Figure 7a: Synchronous Module

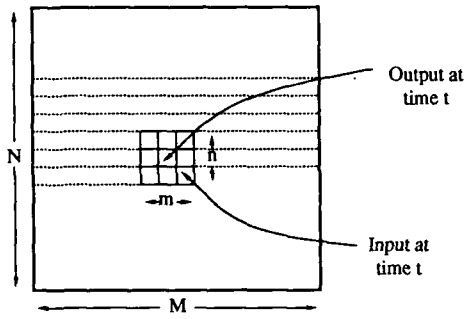



Figure 7b: Buffer belay for mxn neighborhood operation

It is assumed that input data samples arrive sequentially, in raster scan order, and that output image samples are generated and transmitted in raster sequence. In the case of LHR operations, output samples are matched to input samples, so that computations can proceed even as image data arrives. This is not always the case. In non-LHR operations, such as image warp, a given output sample may depend on arbitrary input samples, and it is necessary to wait for the entire input image to arrive before the computation of output samples can begin.

A module includes buffers for each of its input ports. The buffers serve two purposes: to accumulate image data as required to provide the support for computing output samples, and to help align data streams arriving over different paths. The minimal buffer size depends on the operations being performed. In the case of a point operation, no buffer is required. An LHR operation with neighborhood size $n$ by $m$ requires a buffer of size $(n-1)M+m$ samples, where $M$ is the length of an image row, Figure 7b. A non-LHR operation generally requires an input buffer the size of the entire image.

In addition to the operation it performs, a module is characterized by the 'delay' incurred as image data flows from its input to output ports. If we assume the module performs its computations fast enough to keep up with input data, then the delay is dominated by the time between when a given input sample arrives and the time the neighborhood is centered over that sample, and the corresponding output can be computed. As shown in Figure 7b, this is roughly half the minimum buffer size, times $\tau$, the time between successive input samples: $t = (\frac{n-1}{2}M + m - 1)\frac{\tau}{2}$. This delay is a critical factor in designing an efficient system.

The module shown in Figure 7a is assumed to run synchronously with other modules within a processing network. For some applications asynchronous modules are preferred. Each module is then provided with an output buffer as well as its input buffers. Data flow through the network is controlled locally by individual modules and transmission links. A module computes an output sample when (a) requisite input data is in the input buffer, and (b) there is room in the output buffer for the result. A transmission link transfers data when (a) there is data ready in the output buffer of the sending module and (b) there is room in the input buffer of the receiving module. Transfers are coordinated through handshake protocols.

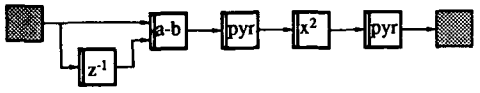
Other control information may travel with the images themselves, indicating, for example, image dimensions, or even the sequence of operations to be performed.

Large system can be assembled from synchronous or asynchronous modules. With asynchronous modules this is simply a matter of linking appropriate units; there is no need for a supervisor system to synchronize data flow. For synchronous modules, data rates need to be matched carefully, but individual modules are considerably less complex than in the asynchronous design. Such systems can be upgraded readily by replacing individual modules with more powerful modules.

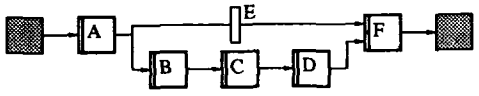
## Lattice Pipeline

Iconic analysis algorithms are implemented by arranging processing modules appropriately in networks. We define a *lattice pipeline* as a network consisting of one or more pipelines running in parallel, possibly with branching and merging paths, but in which there are no loops. [20][21]. Thus a lattice pipeline may be represented topologically as a directed graph, with modules as nodes, and data paths as links. The term 'lattice' is used here in the mathematical sense of a partial ordering: given any two nodes, one node may be said to be greater than the other (upstream on a common data path) or they are unrelated (on different data paths). The lattice pipeline architecture is a direct translation of image flow diagrams, such as those in Figure 2, into physical devices that perform the indicated computations. For example, Figure 8a shows a network topology to implement the change energy computations of Figure 2a.

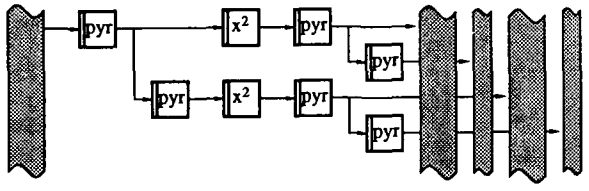
To operate efficiently a lattice pipeline must be 'balanced'. That is, the total delay along all paths between any two processing modules must be (roughly) equal. Figure 8b illustrates the importance of balancing, as well as the means




8a: Lattice Pipeline Realization of change energy computation




8b: Balance the Lattice Pipeline by inserting delay 'E'




8c: Continuous flow-through processing of a "push broom" image.

by which it can be achieved. Here module $A$ sends images to module $F$ along two pathways. One path is direct, without processing, while the other is by way of processing modules $B$, $C$ and $D$. A 'delay element' $E$ is inserted on the direct path to provide balance. $E$ is a buffer that accumulates and retransmits data samples after a prescribed delay. If $B$, $C$ and $D$ have delays $t_b$, $t_c$ and $t_d$ then the delay of $E$ should be (roughly) $t_e = t_b + t_c + t_d$. Without $E$ in the network, the process can deadlock. In an asynchronous design or misregistered synchronous design deadlock occurs when the direct path input buffer in $F$ becomes full, and $A$ must stop transmitting before it has sent sufficient data on the indirect path to allow all of the modules $B$, $C$ and $D$ to start processing.

A network is balanced when the minimum delay on any path between two modules is less than the maximum delay on all other paths between the given modules. A pipeline network can always be balanced through the addition of appropriate delay elements.

The significance of the lattice pipeline design is that relatively simple modules can be configured in networks to perform complex iconic algorithms very rapidly. Tasks such as the generation of texture or motion energy are particularly well suited to this design. A lattice pipeline provides minimum delay between system input and final output, and is therefore to be preferred for real time applications in which system responsiveness is important.

Analysis based only on LHR operations is particularly powerful. Since the delay incurred at each element represents only a few rows of the input image, a given image may be undergoing a sequence of operations simultaneously, and final results may emerge for the beginning rows of an image while the later rows are still being scanned from a camera. Indeed the lattice pipeline may be essential when analyzing certain very large images, such as those obtained by lineal push broom sensors in satellite surveillance and in some industrial inspection tasks. Figure 8c. It is impractical in such cases to store intermediate images; final results must be obtained in flow-through fashion, as image data is being sensed.

On the other hand the lattice pipeline design often does not provide the most efficient use of available computing resources. Due to subsampling and the use of restricted analysis windows, the data rates along some pathways will be significantly less than along others. Thus while some modules run at full capacity, others may run well below their maximum capacity.

### Circulating Pipeline

An alternative architecture is shown in Figure 9a. Here it is assumed that a system consists of a large pool of processing modules, another pool of memory modules, and an interconnection network. Modules are the same as in the lattice pipeline, but data flow is quite different. It is assumed that an iconic operation does not begin until all required input image data resides in memory. The output of the operation is returned directly to memory. Thus a given image cannot be undergoing several sequential operations simultaneously, as in the lattice pipeline. Computations are performed as images move from memory to processor to memory.

The processing algorithms run on a circulating pipeline can be identical to those on a lattice pipeline. Indeed it is often expedient to implement 'virtual' lattice pipelines on such a machine. The advantage of the circulating pipeline is that processing modules can be used very efficiently. There

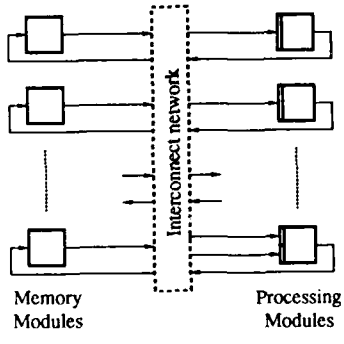



Figure 9a: Circulating Pipeline

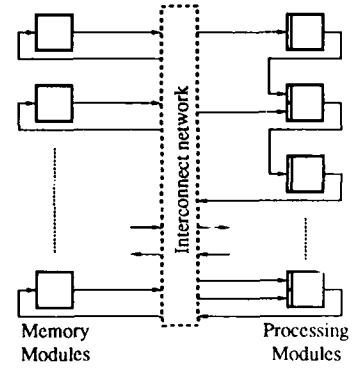



Figure 9b: Hybrid Design

is a cost however, in increased memory as well as in increased overall delay. Since a module does not begin a processing step until all input data is ready, the module can proceed at the full speed when it does begin, and will complete the task in time proportional to the number of samples processed. The module is then free to take part in another task. This system can support multiple independent processes concurrently in order to make full use of computing resources. Scheduling must be coordinated by a system supervisor, but is relatively simple with flow-through processing.

Figure 9b shows a hybrid design, combining features of both lattice and circulating pipeline architectures. With the hybrid design, advantages of both system can be realized. This will be typical of actual machines built for hierarchical image analysis.

## VII. AN IMPLEMENTATION: THE PVM

We are developing several machines to perform multiresolution image analysis. As was indicated in the Section II overview of DSRC research, machines are currently being considered for machine vision, surveillance, and motion analysis. Two quite different designs were completed in 1985. One was built as a lattice pipeline for the processing and enhancement of television signals [5], while the other was developed as a hybrid machine for vision applications [38]. The latter system, called the Pyramid Vision Machine, or PVM1, has been used to demonstrate the basic elements of smart sensing, as outlined above, in Section III. The machine has confirmed that these techniques can provided considerable power even to relatively modest vision systems. The approach has been shown to be very appropriate for such applications as surveillance, industrial inspection and object tracking and recognition.

The PVM1 is shown schematically in Figure 10a. The machine itself is shown in Figure 11. It consists of three (512 by 512) memory modules and two clustered processing modules, interconnected by a limited switching network. One clustered module consists of a multiplier, an ALU and a shifter. The second is a custom built component for performing pyramid operations at video rates. Not shown is the host computer. This controls data flow and computations, and performs higher level vision functions. The host can access data in any of the frame stores. While the PVM generates various feature images rapidly, in a fine-to-coarse fashion, the host can explore this data and locate information of interest through coarse-to-fine search strategies. The PVM has been run with several different host, but most development has been with an IBM/AT.

The PVM1 is designed to run continuously, in real time, on images directly from a video camera. It is capable of constructing full Gaussian and Laplacian pyramids for 256 by 240 image frames at the rate of 30 per second. The pyramid module performs a five by five separable convolution, and handles boarders (very important in a multiresolution system). The pyramid module includes its own microprogrammable controller, as well as look up tables for performing point operations on pyramid levels as they are generated.

To date 9 PVM1 systems have been built. Several can be interconnected to achieve enhanced processing speed.

A second generation pyramid machine, the PVM2, is now under development. The basic system is shown in Figure 10b. This system includes numerous enhancements when compared to the earlier PVM1. In addition to the increased number of memory modules and processors, the system has increased speed, precision, and interconnection flexibility, and can handle large format images. Components of this system include VLSI integration, and are intended to be modular, so that they may be used in other configurations within larger systems.

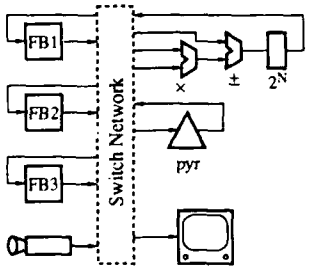


Figure 10a: PVM1

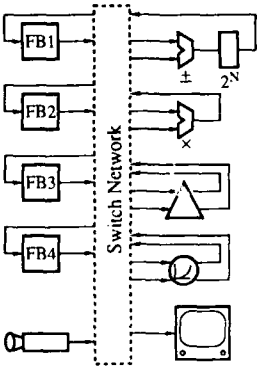


Figure 10b: PVM2


Figure 11. PVM1

## ACKNOWLEDGMENTS

A large number of people have worked together on the research projects reported here. The vision group at DSRC includes Jim Arbeiter, Jim Bergen, Roger Bessler, Peter Burt, Curt Carlson, Matt Clark, Mike Drews, Raj Hingorani, Ray Kolczynski, Al Pica, Joe Sinniger, and Gooitzen van der Wal. Much is owed to Ted Adelson, Charlie Anderson and Joan Ogden who were until recently also group members. The group developing the Princeton Engine includes Steve Bernard, Danny Chin, Stan Knight, Joe Passe and Herb Taylor. The group developing neural networks include Jack Gelfand, John Pearson, and Rick Peterson.

## REFERENCES

1. Adelson, E. H., and J. A. Movshon, 'Phenomenal coherence of moving gratings,' *Nature*, **200**, pp. 523-525, 1982.
2. Adelson, E. H., and J. R. Bergen, 'Spatiotemporal energy models for perception of motion,' *J. Opt. Soc. Am. A*, **2**, pp. 284-299, 1985.
3. Adelson, E. H., E. Simoncelli, and R. Hingorani, 'Orthogonal pyramid transforms for image coding,' *Proc. SPIE Conf. on Visual Communication and Image Process*, Cambridge, 1987.
4. Anderson, C. H., P. J. Burt, and G. S. van der Wal, 'Change detection and tracking using pyramid transform techniques,' *Proc. SPIE Conference on Intelligent Robots and Computer Vision*, Boston, pp. 72-78, 1985.
5. Arbeiter, J. H. and R. F. Bessler, 'A two-dimensional real-time video pyramid processor,' *RCA Review*, **47**, pp. 3-31, 1986.
6. Bergen, J. R., and E. H. Adelson, 'Visual texture segmentation based on energy measures,' *J. Opt. Soc. Am. A*. **3**. p. 98, 1986.
7. Bergen, J. R., and E. H. Adelson, 'Hierarchical, computationally efficient motion estimation algorithm,' *J. Opt. Soc. Am. A*, **4**, pp. 35, 1987.
8. Bergen, J. R., and B. Julesz, 'Parallel versus serial processing in rapid pattern discrimination,' *Nature*, **303**, pp. 696-698, 1983.
9. Bergen, J. R., and B. Julesz, 'Rapid discrimination of visual patterns,' *IEEE Trans. on Systems, Man, and Cybernetics*, **13**, pp. 856-863, 1983.
10. Burt, P. J., 'Fast filter transforms for image processing,' *Comp. Graphics and Image Processing*, **16**, pp. 20-51, 1981.
11. Burt, P. J., 'Fast algorithms for estimating local image properties,' *Comp. Vision, Graphics, and Image Processing*, **21**, pp. 368-382, 1983.
12. Burt, P. J., "Smart sensing' in machine vision,' *Machine Vision*, H. Freeman, Ed., Academic Press, to appear, 1987.
13. Burt, P. J., 'Smart sensing in machine vision,' IEEE Proc. (submitted) 1988.
14. Burt, P. J., 'Moment images, polynomial fit filters, and the problem of surface interpolation,' submitted 1988.
15. Burt, P. J., and E. H. Adelson, 'The Laplacian pyramid as a compact image code,' *IEEE Trans. Comm.*, **31**, pp. 532-540, 1983a.
16. Burt, P. J., and E. H. Adelson, 'A multiresolution spline with application to image mosaics,' *ACM Transaction on Graphics*, **2**, pp. 217-236, 1983b.
17. Burt, P. J., C. H. Anderson, J. O. Sinniger and G. van der Wal, 'A pipeline pyramid machine,' in *Pyramidal Systems for Computer Vision*, Cantoni, Levialdi, Eds., NATO ASI Series ARW, 1986.
18. Burt, P. J., and B. Julesz, 'A disparity gradient limit for binocular fusion,' *Science*, **208**, pp. 615-617, 1980.
19. Burt, P. J., and G. Sperling, 'Time, distance and feature trade-off in visual apparent motion,' *Psychological Review*, **88**, pp. 171-195, 1981.
20. Burt, P. J., and G. van der Wal, 'Iconic image analysis with the Pyramid Vision Machine (PVM),' *IEEE Workshop on Computer Architecture for Pattern Analysis and Machine Intelligence*, Seattle, 1987.
21. Burt, P. J., and G. van der Wal, 'Architectures for hierarchical iconic image analysis,' IEEE PAMI (submitted), 1988.
22. Burt, P. J., C. Yen, and X. Xu, 'Multiresolution flow-through motion analysis,' *IEEE Computer Vision and Pattern Recognition Conference Proceedings*, Washington, D.C., pp. 246-252, 1983.
23. Carlson, C. R., and R. W. Cohen, 'A simple psychophysical model for predicting the visibility of displayed information,' *Proceedings of the SID*, **21**, pp. 229-246, 1980.
24. Carlson, C. R., R. W. Klopfenstein, and C. H. Anderson, 'Spatially inhomogeneous scaled transforms for vision and pattern recognition,' *Opt. Letter.*, **6**, pp. 386-388, 1981.
25. Chin, D., J. Passe, F. Bernard, H. Taylor, and S. Knight, 'The Princeton Engine: a real-time video system simulator,' in preparation, 1988.
26. Daubechies, I., 'Orthogonal basis of compactly supported wavelets,' in press, 1988.
27. Glimm, J., and A. Jaffe, 'Quantum physics: a functional point of view,' *Springer*, New York, 1981.
28. Goupillaud, P., A. Grossmann, and J. Morlet, 'Cycle-octave and related transforms in seismic signal analysis,' *Geoexploration*, **23**, pp. 85, 1986.
29. Klopfenstein, R. W., and C. R. Carlson, 'Theory of shape-invariant imaging systems,' *Journal of the Optical Society of America A*, **1**, pp. 1040-1053, 1984.
30. Larkin, L. I. and P. J. Burt, 'Multiresolution texture energy measures,' *Computer Vision and Pattern Recognition Conference Proc.*, Washington, D. C., pp. 529-520, 1983.
31. Lee, W. A., 'Coring on oriented band passed images,' Senior Thesis, Princeton University, 1988.
32. Mallat, S. G., 'A compact multiresolution representation: The wavelet model,' Computer and Information Science, Univ. of Pennsylvania, MS-CIS-87-69, 1987.

33. Pearson, J. C., W. E. Sullivan, J. J. Gelfand, and R. M. Peterson, 'A computational map approach to sensory fusion,' *Proc. AOG/AAAIC 87 Joint Conf. on Merging Tomorrow's Technology with Defense Readiness Requirements*, 1987.
34. Pearson, J. C., L. H. Finkel, and G. M. Edelman, 'Plasticity in the organization of adult cerebral cortical maps: a computer simulation based on neural group selection,' *J. of Neuroscience*, **7**, 1987.
35. Rose, A., 'The sensitivity performance of human eye on an absolute scale,' *J. Opt. Soc. Am.*, **38**, pp. 196, 1948.
36. Schade, O. J., Sr., 'Optical and photoelectric analog of the eye,' *J. Opt. Soc. Am.*, **46**, pp. 721, 1956.
37. Silverman, D. J., G. L. Shaw, and J. C. Pearson, 'Associative recall properties of the trion model of cortical organization,' *Biological Cybernetics*, **53**, pp. 259-271, 1986.
38. van der Wal, G. S., and J. O. Sinniger, 'Real time pyramid transform architecture,' *SPIE Proceedings Intelligent Robots and Comp. Vision*, Boston, pp. 300-305, 1985.
39. van Essen, D. C., and C. H. Anderson, 'Sampling of the visual image in the retina and LGN of the macague: a quantitative model,' *Investigative Opthalmology and Vision Science*, **27**, pp. 94, 1986.
40. Wilson, H. R. and J. R. Bergen, 'A four mechanism model for threshold spatial vision,' *Vision Res.*, **19**, pp. 19-32, 1979.

# THE MARYLAND APPROACH TO IMAGE UNDERSTANDING

John (Yiannis) Aloimonos
Larry S. Davis
Azriel Rosenfeld

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3411

## ABSTRACT

In our efforts to understand images, while we still work on initial processes of low and middle level vision, emphasis is being placed on the *integration of multiple sources of information* for visual reconstruction, on *navigation* and on *object recognition*. We introduce a new methodological paradigm for research in vision, namely: while we continue our research top-down in the Marr paradigm, we can also work in a bottom-up fashion in that paradigm. We also suggest that the Marr paradigm (computational theory, algorithms, data structures, and implementation) should be augmented with one more level, that of *robustness*, that Marr left implicit in his writings.

## 1. INTRODUCTION

In this paper we describe some results of our recent and current research. We analyze many low, middle and high level vision problems, from edge detection and "shape from $x$" to navigation and recognition of objects. Our research is focused on visual abilities rather than on the construction of entire systems, i.e., systems that exhibit some vertical integration and use knowledge at all levels including domain specific information. "In order to complete the construction of such systems, it is almost inevitable that corners be cut and many oversimplified assumptions be made" [Brady, 1982]. Such systems might be able to carry out a limited number of operations (possibly beneficial in simple industrial situations), but they do not enhance our understanding of vision in general.

Due to the inability of the general theories developed up to now in the Marr paradigm (shape from $x$, optic flow, structure from motion) to find applications in practical systems (such as recognition systems, navigation, etc.), some researchers are devoting their efforts to developing systems. We feel that such an approach should be conducted in the spirit of the Marr paradigm.

To be more explicit, a large part of modern computer vision develops theories for the explanation of visual abilities, in the spirit of the Marr paradigm, i.e., before we do anything we must understand what we want to compute and why, and develop a computational theory for the problem at hand (develop constraints between the desired unknowns and the data). Then we should design the algorithm that will execute that particular process, design the required data structures, and finally implement the algorithm on a machine. If all three levels are well understood then we can talk about understanding that particular visual process. So, a part of modern computer vision works top-down in the Marr paradigm (see Figure 1). When, in this paradigm, the theories of shape from $x$, structure from motion, projective invariance, and so on, have been highly developed, then the solution to the hard problems of object recognition, navigation, visual learning, etc., may be easily determined. On the other hand, one who wishes to develop visual systems should perceive the problem as finding *a general solution* to *a specific problem*, as opposed to finding a *specific solution* to a *general problem*—classical Marr paradigm research, such as shape from $x$, etc. Indeed, one does not need (for example) to recognize the objects in the extrapersonal space in order to navigate. Simple collision avoidance and homing procedures would be adequate. One does not necessarily need "structure from motion" or "stereo" in order to avoid obstacles. There may be other processes for accomplishing such a task. But in building a visual system for navigation, for example, we should do the analysis in the spirit of the Marr paradigm, by developing a computational theory for the task. The only difference now is that we are, in a sense, working bottom-up in the Marr paradigm (Figure 1). In addition, a theoretical stability analysis of the algorithmic level (implicit in Marr's writings) is becoming explicit today by necessity, because excellent computational theories result in unstable algorithms [Tsai and Huang, 1984].

An example will help to clarify matters. Extracting information from images (shape from $x$, for example, which aims to *reconstruct* the visible world) is a lot of work. The visual cortexes of animals which carry out complex visually moderated behaviors, contain millions of neurons, each of which performs computations which by the lowest estimates require thousands of computer steps per second to simulate. Much of this capacity is probably necessary to carry out cortical image processing. Thus, extracting shape from $x$ (reconstruction) is hard and although recent literature has shown excellent results, few of the published theories have found practical applications. Motivated by this, recognitionists propose solving problems, specific problems.

The reason for the limited success obtained by the reconstruction school is either that the theories are not well developed yet or that the fundamental approach is overambitious. This second reason is a result of the tendency to

The support of the Defense Advanced Research Projects Agency and the Center for Night Vision and Electro Optics under Contract DAAB07-86-K-F073 is gratefully acknowledged, as is the help of Sandy German in preparing this paper.

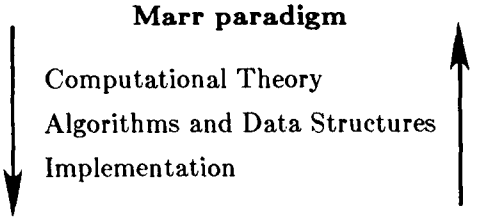



**Figure 1**: Shape from $x$ or $x$ from $y$ research comes under the top-down Marr paradigm. Research for developing a system for solving a particular problem should come under the bottom-up Marr paradigm.

attempt to reduce multiple visual tasks to some common denominator (a set of visual modules)—something established in the Marr paradigm and indeed the scientific way of approaching the problem. For example, the problem of visual navigation has often been considered a subproblem of the "shape from $x$" problem and it is mentioned as an application of the more general studies designed to extract accurate, quantitative information about the three-dimensional structure of the world. However, despite the numerous elegant theoretical results, none of the shape from $x$ studies have yielded systems which can actually be used to navigate in a real environment. The usual situation is that a mathematical solution to the general problem can be obtained only under specific assumptions which are not particularly valid in the real world. On the other hand, recognitionists who have been working with practical applications have been so concerned with performing one narrowly defined operation as to render their results devoid of useful general principles. We feel that there is another approach, between reconstruction and recognition. While doing research on obtaining a specific solution to a general problem (i.e., shape from $x$), one may seek a general solution to a specific problem, for example, solving the visual obstacle avoidance problem with no specific restrictions on the shape of the environment. In this approach, qualitative measurements which are specific to the problem at hand, but which can be obtained practically in real world environments, can be used. Such recognitionist research can be classified in the bottom-up Marr paradigm, and if there is a module in the human visual system that performs this particular task, then the recognition and reconstruction schools are no different. We know that there is no module in the human visual system that recognizes yellow Volkswagens, but modules may well exist for tasks such as obstacle avoidance, visual stabilization, homing, navigation, and the like. In the sequel we will describe our work on the low, middle and high level aspects of vision.

## 2. EARLY (LOW LEVEL) VISION

### 2.1. Discontinuous Regularization

Most of the early vision problems are ill posed in the sense of Hadamard. Poggio and his colleagues suggested the theory of regularization as a possible solution (in a common paradigm) to such problems. It means that we find the solution that is as consistent as possible with the data and we also require that the solution be smooth. One can show that the problems of shape from shading, shape from texture, shape from motion, stereo, optic flow computation, and edge detection are ill-posed and for some of them a unique smooth solution exists. Let us consider, as an illustration, the shape-from-$x$ problems (texture, shading, stereo, motion).

In all the cases of shape from $x$ (with the exception of shape from contour), the constraint relating the gradient $(p,q)$ of a surface patch to the data at the point $(x,y)$ which is the projection of that patch is of the form

$$L(p,q,x,y) = A(x,y)p^2 + B(x,y)q^2 + C(x,y)pq + D(x,y)p + E(x,y)q + F(x,y) = 0$$

where $A$, $B$, $C$, $D$, $E$, $F$ are functions of the position in the image and depend on the particular physical parameters (data). The idea of solving ill-posed problems such as the above, i.e., restoring "well-posedness", is to introduce suitable a priori knowledge that will restrict the space of admissible solutions. The assumption introduced by Poggio is that of smoothness. The problem of finding $(p,q)$ at every image point then reduces to finding the minimum of

$$\int_\Omega\int L^2(p,q,x,y) + \lambda(p_x^2 + p_y^2 + q_x^2 + q_y^2)dxdy$$

We are faced immediately with the following problems:

(a) The constraint $L = 0$ is nonlinear, and standard regularization deals with linear constraints. The situation is different if $L$ is convex, but it is not for the above shape from $x$ cases, with the exception of the stereo case.

(b) Surfaces are not everywhere smooth; they are smooth only in between discontinuities. If we require smoothness as the a priori assumption, the solution we will obtain will have been smoothed over discontinuities and so it will not be correct.

(c) Even if we incorporate smoothness, why use the Dirichlet norm $\int\int (p_x^2 + q_x^2 + p_y^2 + q_y^2)dxdy$? Why not second order derivatives or higher exponents? And what determines the optimal $\lambda$?

In [Aloimonos and Shulman, 1987] we answered the first question. It is always possible to get a unique smooth solution, but one may have to oversmooth.

To answer the second question one would have to develop a theory of regularization in the presence of discontinuities. Considering this problem very interesting (since its solution will advance other fields too), we have been working toward the development of such a theory. Up to now, we have developed a linear theory of discontinuous regularization, and we are currently working on its nonlinear extension. In the rest of this section we will briefly describe our theory but we will apply it to nonlinear problems (shape-from-$x$) so that the existing problems for the nonlinear theory will become clear.

Several approaches to regularization in the presence of discontinuities have been discussed by computer vision researchers. Many of these approaches require that a binary distinction be made between points that are and points that are not discontinuity points. However, even if a point is not a discontinuity point, if the data provide evidence that there is a sharp change in orientation in the vicinity of that point, then we do not want the smoothing term in our regularization condition to force us to smooth over the sharp orientation

change. Many of the existing discontinuous regularization techniques require that a segmentation of the image be performed. But segmentation is a difficult problem; there is no rigorous theory of segmentation. One of the reasons we want to know the shape of an object is to facilitate segmentation, and if we are to do explicit segmentation, we should learn how to segment. Existing techniques do not facilitate automatic learning of segmentation. Other discontinuous regularization methods involve complex non-linear equations that are solved stochastically. This theory is simpler and completely deterministic, and it has already been applied to the nonrigid motion case [Shulman and Aloimonos, 1988].

The basic insight is that we can expect the errors of nearby points to be correlated. Thus $\partial L/\partial x$ and $\partial L/\partial y$ should be small. The reasons errors of nearby points are correlated are many One reason is that the errors in determining the light intensities of nearby points are correlated and not all of this correlation is removed by preprocessing. Another factor is that a quadratic measure of smoothness such as $\int\int\left[(f_x)^2+(f_y)^2+(g_x)^2+(g_y)^2\right]$ excessively penalizes large changes in orientation. This measure of smoothness implicitly assumes that the variables $f_x$, $f_y$, $g_x$, $g_y$ have normal distributions. A more correct smoothness measure would allow the derivatives of $f$ and $g$ to have probability distributions with bigger tails. In order to correct for an incorrect smoothness measure, we have to force the derivatives of $L$ to be small. (The usual regularization condition produces extremely large jumps in the value of $L$ in the vicinity of discontinuities.) Another way to think of discontinuous regularization is that not only must the solution be consistent with the data but the derivatives of the solution must also be consistent with the derivatives of the data.

In general, a variational condition in discontinuous regularization theory takes the form

$$\text{minimize} \int\int\left[\sum_0^\infty a_{ij}(\partial^{i+j}L/\partial x^i\partial y^j)\right]^2 dxdy$$

$$+\int\int\sum_0^\infty b_{ij}(\partial^{i+j}f/\partial x^i\partial y^j)^2$$

$$+\int\int\sum_0^\infty c_{ij}(\partial^{i+j}g/\partial x^i\partial y^j)^2.$$

Here the $a_{ij}$, $b_{ij}$, $c_{ij}$ are constant parameters. The first integral involves a summation from 0 to $\infty$ because it is arbitrary to require only the first order derivatives of $L$ to be small and not to impose a requirement on the higher order derivatives. Similarly, the smoothness condition (i.e., the second and third integrals) must also involve a summation from 0 to $\infty$. Some of the coefficients $a_{ij}$, $b_{ij}$, $c_{ij}$ can be 0. Because of the noisiness of higher-order derivative estimates, the coefficients $a_{ij}$ should rapidly approach 0 as $i,j\to\infty$.

We do still have a problem because computing the values of derivatives of $L$ requires calculation of the values of derivatives of the data and that calculation is numerically unstable. As Poggio suggests, in order to differentiate numerically (and numerical differentiation enables us to do local edge detection), we apply the theory of regularization. This theory of numerical differentiation is different from Poggio's because it is concerned about preserving discontinuities in the derivatives. Indeed, if we are given the value of a function $h$ at a finite number of points $P_1$, $P_2$, $P_3$ $\cdots$ $P_n$ and we know these values are corrupted by noise, then the optimal approximation to the derivative of $h$ need not be a linear function of the values $h(P_1)$, $h(P_2)$ $\cdots$ $h(P_n)$. This is true despite the fact that $\partial h/\partial x$ is a linear functional of $h$. However, because linear learning is simpler than nonlinear learning, we will temporarily restrict ourselves in this paper to linear approximations to the derivatives. Thus, our estimates of derivatives should be very much like Poggio's. But we do not have to actually compute these derivatives. All we need to know is that the derivatives of $L$ will be approximated by linear functionals of $L$ and we can learn which linear functionals to use. Thus the first integral of our variational condition is approximated by an expression of the form $(AL)^2$ where $A$ is a matrix that has to be learned. The second and third integrals are approximated by a polynomial that is quadratic in the variables $f_{ij}$, $g_{ij}$, which represent the orientations at the data points $(i,j)$. This procedure would result in nonlinear learning, which is not well understood yet. There is a better way to obtain a linear learning problem and that involves first learning how to compute derivatives of intensity. Once we know how to do that, we know how to compute the derivatives of the coefficients that appear in the formula for $L$ and thus we can compute derivatives of $L$. We will obtain an Euler-Lagrange equation of the form

$$A(L)\frac{\partial A(L)}{\partial f}=-\Phi_1\xi$$

$$A(L)\frac{\partial A(L)}{\partial g}=-\Phi_2\xi$$

where $A(L)$ is a matrix obtained by applying a possibly nonlinear functional to the matrix $L$. In fact, $A(L)$ is a sum of the form $\sum a_{ij}A^{ij}(L)$ where the $a_{ij}$ are constant coefficients and the $A^{ij}(L)$ are approximations to the derivatives $\partial^{i+j}L/\partial x^i\partial y^j$. Thus finally we obtain

$$\sum a_{ij}^2A^{ij}(L)\partial A^{ij}(L)/\partial f=\Phi_1\xi$$

$$\sum a_{ij}^2A^{ij}(L)\partial A^{ij}(L)/\partial g=\Phi_2\xi$$

with the $A^{ij}(L)$ being functions that we know how to compute. Write $\phi_{ij}(\xi)$ for the known function $\begin{Bmatrix}A^{ij}(L)\partial A^{ij}(L)/\partial f\\ A^{ij}(L)\partial A^{ij}(L)/\partial g\end{Bmatrix}$; then we have

$$\sum a_{ij}^2\phi_{ij}(\xi)=\Phi\xi$$

where the $a_{ij}^2$ and $\Phi$ have to be learned. We will impose additional constraints to insure that the values of $(a_{ij}^2,\Phi)$ are unique. We still have to describe how to compute $A^{ij}(L)$ (i.e., how to compute derivatives of intensity). If we assume that we are only able to do linear learning this is simple: $\partial E/\partial x$, $\partial E/\partial y$, etc. are linear functions of $E$ so we have equations such as $\partial E/\partial x=AE$ and we have to learn the matrix $A$ that approximates the derivative. A priori, we expect that $A$ will be obtained by applying a smoothing filter (such as a Gaussian) to a weighted sum of difference coefficients $\Delta E/\Delta X$, $\Delta^2E/\Delta X^2$, $\Delta^3E/\Delta X^3$, etc.

To summarize, this theory of discontinuous regularization requires us to impose constraints on the derivatives of the "error" $L$. This leads to equations of the form $\sum a_{ij}^2\phi_{ij}(\xi)=\Phi\xi$ where we need to learn $a_{ij}^2$, $\Phi$. More generally, we could let the $a_{ij}^2$ be matrices rather than constants

(the constraints that the derivatives of $L$ be small could be relaxed at certain points). We must find a procedure for learning the matrices $a_{ij}^2$, $\Phi$ that is simple, robust, implementable in a neural net and leads to a unique solution.

Being biased in favor of sparseness, we want $a_{ij}^2$ to be near zero if $i,j$ are large and $a_{00}$ to be nearly the identity matrix. If $a_{00} = I$ and $a_{ij} = 0$ for $i,j > 0$ then we obtain the equation

$$L\,\partial L/\partial g = -\Phi_1 \xi$$

$$L\,\partial L/\partial g = -\Phi_2 \xi$$

which is the usual nondiscontinuous regularization condition. Rewrite $\sum a_{ij}^2 \phi_{ij}(\xi) = -\Phi\xi$ as

$$\left(\sum \hat{a}_{ij}^2 \phi_{ij}(\xi)\right) + \Phi\xi = -\phi_{00}(\xi)$$

where $a_{00} = I$ and $\hat{a}_{ij}^2 = a_{ij}$ otherwise. Thus the first term in the equation represents the discontinuous correction to the usual regularization condition and we want this term to be as small as possible. To make it easier to work with, rewrite the previous equation in the form

$$\Gamma\,\Xi = -\phi_{00}(\xi)$$

where $\Xi$ is the vector

$$\begin{pmatrix} \phi_{00}(\xi) \\ \phi_{01}(\xi) \\ \phi_{10}(\xi) \\ \vdots \\ \phi_{mn}(\xi) \\ \xi \end{pmatrix}$$

and $\Gamma$ is a *matrix to be learned*.[1] The same questions we ask about $\Phi$, we now ask about $\Gamma$: what do we do if the examples give *incomplete information* or *inconsistent information* about $\Gamma$?

The $\Gamma$ we choose is the least squares solution. Under reasonable and quite general statistical assumptions, the least squares solution is the most probable value for $\Gamma$. Computing the least squares solution involves calculating the Moore-Penrose inverse. This procedure is optimal in the sense that it produces as accurate an estimate of $\Gamma$ as we can obtain from our limited set of examples, but it requires a neural net to store large matrices and make very complex calculations. So instead of searching for the optimal solution of the above equation we can calculate the best solution within a restricted subspace. The procedure for calculating this restricted subspace solution is very similar to that for calculating the full space solution but the computational complexity is much less and the speed of convergence is much greater. Another technique is the Robbins-Monro procedure. This results in slower convergence but the calculations are very easy to implement on a neural net. Whatever procedure we use, it may take too many examples to learn $\Gamma$. We may use a priori knowledge to accelerate convergence. For example, we know that $\Gamma$ varies smoothly as a function of position (and so we can use regularization). The $\Gamma$ we use is partly determined by examples and partly by a priori knowledge. As we acquire more examples, the influence of a priori knowledge on our estimate of $\Gamma$ decreases.

Before going on, let us review what learning (rather than stipulating a priori) the value of $\Gamma$ will accomplish for us. $\Gamma$ hides the regularization parameter $\lambda$. This parameter might be allowed to vary from place to place. (We might require a different amount of smoothing near the boundary than at the center of the visual field.) Our smoothing condition might involve first, second, or higher-order derivatives or some linear combination of derivatives of different orders. $\Gamma$ weights the relative importance of the derivatives of $L$ being small. This can vary with position. By varying $\Gamma$, we can take into account all these possibilities. We do not know which of these possibilities is correct. That is why we want a general *learning theory to help us find the proper* $\Gamma$ (to find the least squares solution of the previous equation).

After $\Gamma$ is learned, then we have the "actual" constraint for shape from $x$ problems. If we then solve this equation (nonlinear in shape) we can compute shape from $x$. How one can solve the resulting equation with a method that works all the time is not known yet. Maybe the learning of $\Gamma$ can be done in such a way that the solving of the equation will be easy. All these considerations constitute important research topics.

When all this is understood (and much research is required for it), then research will turn to learning the constraints themselves. According to what was previously discussed, the constraints for shape from $x$ can be learned in the framework of discontinuous regularization. This learning is not from scratch (lately some researchers are trying to solve such problems with learning from scratch) because the basic laws of physics and geometry are first used and then factors such as variational conditions, relative importance of smoothing vs. the constraint, etc., are learned from nature through examples.

### 2.2. Structure from Motion (Dynamic Imagery)

After suspecting (unfortunately without a proof as yet, but from experimentation) that the problem of structure from motion based on point correspondences is unstable, we have put our efforts into solving the problem from correspondences of macro-features (lines, contours, etc.) or without correspondences (based only on the intensity function), and to theoretically analyzing the structure from motion problem (the problem itself may be unstable, no matter what algorithms are used). In Spetsakis and Aloimonos, 1988 we give a closed form solution to the problem of structure form motion from line correspondences (lines not line segments). In Aloimonos and Basu, 1988 we address the problem of structure from motion from the correspondence of planar contours, without point correspondences. Finally, we have found a solution to the structure from motion problem without correspondence, in the general case, using mild assumptions. The situation is as follows. Assume that a camera is moving with no acceleration while continuously acquiring images. The zero crossings

[1] By learning we mean learning from examples (adaptive estimation). If we are given shapes $\Xi_1, \Xi_2, \ldots, \Xi_n$ and their corresponding data (images) $\phi_{ij}(\Xi_1), \phi_{ij}(\Xi_2), \ldots, \phi_{ij}(\Xi_n)$ from these we need to learn the matrix $\Gamma$ $[\Xi_1 \ldots \phi_{ij}(\Xi_n)]$

of all images form surfaces in $xyt$ space, with $t$ being time). The derivatives of those surfaces are related to the instantaneous velocity of the camera through nonlinear equations. Thus, speaking in theoretical terms, the passive navigation problem could be solved without correspondence for inertial motion, *if a particular nonlinear system (as described in* [Ito and Aloimonos, 1988]) could be solved. Our attention has also turned to the robustness analysis of the structure from motion problem, independent of the algorithm used. Maybe the problem is unstable by its nature and should be formulated differently. But to prove that the problem is unstable, regardless of the algorithm used, is not easy. In a sense, one would have to define a probability distribution on the space of all problems, and then compute the probability that a problem is unstable, where instability can be expressed as the distance from a set of ill-posed problems. Such an approach has been initiated in numerical analysis.

In addition, some recent research by Chou and Kanatani [1987] has rigorously analyzed the correspondenceless structure from motion problem for the case of planar surfaces. Finally, we have devised a very robust method for egomotion that will be described in Section 4.1 [Nelson et al., 1987].

### 2.3. Robustness of Visual Algorithms

This kind of research is methodologically important. The literature in recent years has offered a plethora of elegant algorithms that are unstable. There exists a need to enrich the Marr paradigm with the *level of stability*. An unstable algorithm can serve neither as the basis for the explanation of visual capabilities nor as the basis for the construction of intelligent machines. On the other hand, it is hard to study the sensitivity and stability of an algorithm that takes input form nature. One of the reasons is the difficulty of specifying the probability distribution of the noise. When uncertainty is introduced, several known techniques can be used for estimating a bound on the error (in [Aloimonos and Basu, 1988] such techniques are demonstrated ).

The case of evaluating the error due to quantization in computer vision has been extensively studied [Kamgar-Parsi and Kamgar-Parsi, 1987] in our laboratory. Quantization is always necessary in order for the image to be processed by a digital computer, and spatial quantization is not the only kind. The digitization of brightness is important too.

Due to the important role that digitization error plays in the field of computer vision, a careful analysis of its impact on the computational approaches used in the field is called for. Such analysis can serve two purposes: a) It helps us learn how much the results of the computations are affected by this error. b) It can provide insight on how to reduce the impact of this error. Note that unlike most other types of error, digitization error cannot be reduced by performing more accurate experiments. Rather it is caused by the inherent limitations of the instruments used in image processing. Therefore, a careful analysis of digitization error can reveal the intrinsic limitations of a given approach or algorithm.

The theory developed in the above reference can be used to evaluate errors in the estimation of spatial derivatives, calibration, stereo triangulation and the like. This approach yields probability distributions for the required parameters.

## 3. MIDDLE (INTERMEDIATE) VISION

We continue to work on the general integration problem, with the goal of deriving a robust description and explicit representation of the structure of the environment and its properties through the fusion of various early vision algorithms. Although more work needs to be done in the area of early vision algorithms, we have for some time been thinking about the integration of multiple visual information sources in a robust vision system. This stage of visual processing we call intermediate or middle level vision. Here, the problem is to integrate information about the physical processes that underlie imaging, in order to form a unified and robust description. Our theory of discontinuous regularization can be extended to deal with this problem, i.e., that of integrating information from several sources. The extension is not hard and is one of our current research topics. It is worth noting that Poggio and his colleagues are working on the same problem but using a different vocabulary, namely a probabilistic one.

Our research up to now on the interaction of modules, i.e., on the computation of $x$ from $y$ and $z$ (and $w$), can be summarized in Figure 2. On the left of the figure are intrinsic parameters and on the right are cues. We see that an intrinsic parameter is computed from the combination of two or more cues. Some problems that were ill posed now become well-posed, simply because there are more constraints; and some problems that were unstable now become stable, simply because there are more constraints. In [Aloimonos and Basu, 1988] one can find processes for the computation of shape from texture and motion, texture and contour, etc. We are still working on extending the list of results, and as we have already stated, developing the mathematics for this integration (which mathematics will be an extension of the theory of discontinuous regularization).

### 3.1. Active Vision

We now know the problematic aspects of passive monocular vision. Some alternatives have been suggested such as discontinuous regularization and combining information from different sources. While discontinuous regularization is a topic of current research, combining information from several sources seems promising but it suffers from the fact that additional information sources may not be available. The active vision paradigm attempts to account for this problem.

An observer is called active when engaged in some kind of activity whose purpose is to control the geometric parameters of the sensory apparatus. The purpose of the activity is to manipulate the constraints underlying the observed phenomena in order to improve the quality of the perceptual results. For example a monocular observer that moves with a known or unknown motion or a binocular observer that can rotate its eyes and track environmental objects are just two examples of an observer whom we call active. In other words, still staying within the paradigm of combining information from different sources, we create one more information source (the particular activity), by making the observer active. This paradigm is partly motivated by human perception, which is active. Perceptual activity is exploratory and searching. When humans see and understand, they actively look. In the process of looking, their eyes adjust to the level of illumination, focus on certain things, converge or diverge, and their heads move to obtain a better view of the scene.

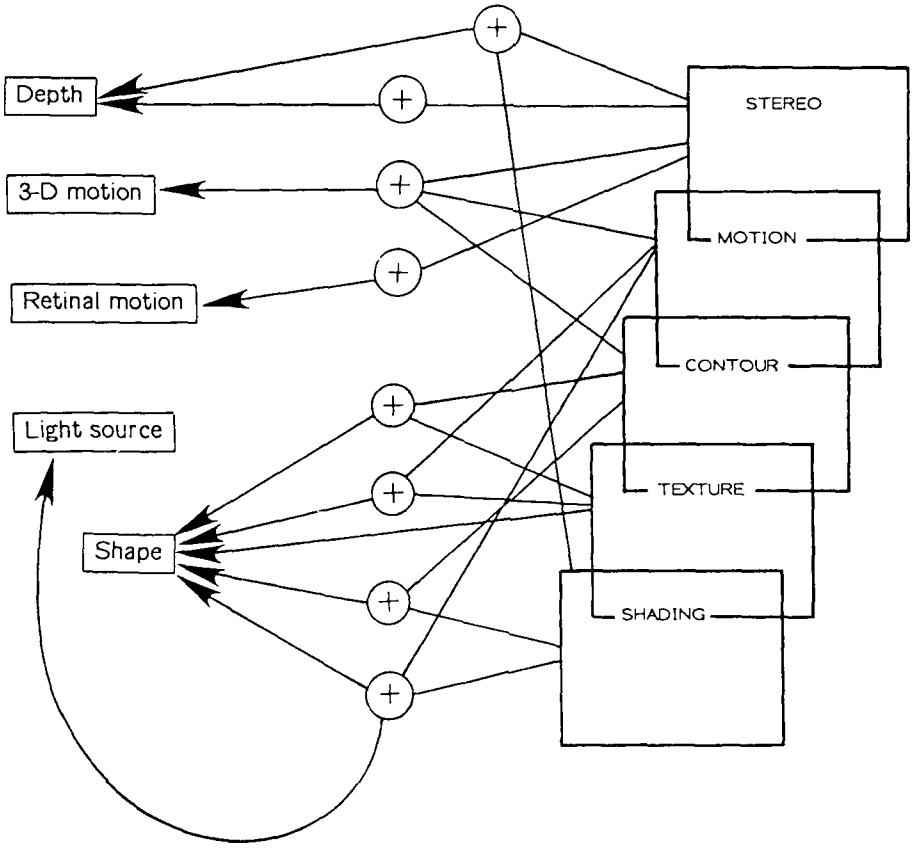



**Figure 2**

One can ask why human observers operate in such a way, because certainly humans are very efficient in visual tasks. In other words, how does the fact that an observer is active affect the levels of a visual system (as described by Marr), namely computational theory, representations and processing algorithms, and implementation? Does an active observer have any advantage over a passive observer, in any computational theoretic, algorithmic, or implementational way?

The activities studied up to now in the case of an active observer include touch and motion [Bajcsy, 1986; Aloimonos et al., 1987], while many of them are unexplored. In the case where the activity is a known motion of the observer (or tracking environmental points and/or converging the cameras), results have been reported [Aloimonos et al., 1987] that state that all the shape from $x$ problems become well-conditioned and unique solutions are possible. This, which should not be surprising, is due to the fact that there is an additional information source, controlled by the observer him/herself.

The basis for the approach lies in being able to work in a rich stimulus domain with a partially known parametrization. This knowledge is due to the fact that the viewing transformation is known. As the viewing parameters are continuously varied, the observed visual stimuli undergo local transformations that are measurable and provide powerful constraints for the computation of the unknown scene parameters. We are, of course, interested in the rates of these stimulus changes, which have traditionally been thought to be difficult to measure. However, it should be pointed out that in the Active Vision (AV) paradigm we do not work with a small set of discrete observations, but with trajectories in the stimulus space, termed flow lines. These trajectories are smooth, since the viewing transformations we use are themselves smooth, and therefore can be computed accurately enough for our purposes. Thus we do not need to rely on the smoothness of properties of the observed scene, such as illumination and depth. The real power of the approach is due to the avoidance of complications usually associated with multiview approaches to visual perception. For instance, the problem of correspondence of microfeatures is not involved in the AV approach.

Table 1 [Aloimonos et al., 1987] compares the performance of a passive and an active observer in the solution of several basic problems.

Finally, work on *active* vision will determine the feasibility of vision systems and will open up research issues of *exploratory* and *feedback* vision. Exploratory vision amounts to finding the activity (from the space of all activities) that will give rise to the most stable algorithm for the task at

Table 1

| Problem | Passive Observer | Active Observer |
|---|---|---|
| Shape from shading | Ill-posed problem. Needs to be regularized. Even then, unique solution is not guaranteed because of nonlinearity. | Well-posed problem. Unique solution. Linear equation. Stability. |
| Shape from contour | Ill-posed problem. Has not been regularized up to now in the Tichonov sense. Solvable under restrictive assumptions. | Well-posed problem. Unique solution for both monocular and binocular observer. |
| Shape from texture | Ill-posed problem. Needs some assumption about the texture. | Well posed problem. No assumption required. |
| Structure from motion | Well posed but unstable. Nonlinear constraints. | Well posed and stable. Quadratic constraints, simple solution methods, stability. |

hand. This can be quantified. For example, given a known motion for the active observer, we can prove that he/she can uniquely compute shape from texture, by solving a linear system. But for what motion (activity) is the solution of the system the most stable? Feedback vision will have to do with how information gathered from the environment can be used to guide future activities. Researchers have begun to implement systems that exhibit active vision characteristics.

## 4. HIGH LEVEL VISION

Here we describe our research in navigation, object recognition and learning.

### 4.1. Navigation

There are two motivations for our study of navigation.

First the practical. There is presently a growing interest in automatic systems capable of adaptive interaction with a physical environment. By adaptive, we mean that the movement of the system is governed by some sort of sensory feedback which allows it to adapt to variation in environmental conditions rather than being limited to a small set of fixed motions as is the case with, for instance, cam activated machinery. Examples of such systems (roughly ordered from the extant to the hypothetical) include applications in automated manufacturing, guided weapons systems, autonomously piloted vehicles, automatic assembly of structures in space, and household robots. All these applications require the system to utilize sensory input to mediate movement, either of the system itself relative to the environment, or of one component of the environment relative to another. In this document, the term navigation refers in a broad sense to such sensory mediated movement. However, the emphasis will be primarily on the movement of the system within the environment rather than on the more complex task of manipulating one part of the environment with respect to another.

There exist a number of automatic systems which perform various forms of navigation quite successfully. Examples include intercontinental ballistic missiles, heat seeking missiles such as the Stinger, commercial aircraft navigation systems which can effectively fly a plane across an ocean and land it without pilot intervention, and celestial navigation systems such as that aboard the Voyager spacecraft. Most of these depend on sensory data other than visual, though the infrared detectors in the Stinger, and the star location system of the Voyager, can be considered as simple visual systems. These systems operate within rather simple, predictable environments, containing few objects besides the system vehicle and perhaps a well specified target. Navigation can consequently be adequately performed on the basis of an inertial system, or from the location of a few stars or radio beacons.

Such limited sources of information do not suffice, however, when the environment contains complex features with which the system must interact. A spacecraft in outer space can steer from point A to point B by the stars. A vehicle following a road cannot. One solution to this problem is to provide the system, beforehand, with a complete, model of the salient features of the environment. The cruise missile uses a system of this type. The disadvantages are, first, that complete models are extremely memory intensive so that a given system can be competent only within a few specific environments, and, second, that the required model may be unavailable even in principle (for instance, a prior model giving the location of all cars on a public highway at a given time is untenable). An alternative is to use a more sophisticated sensory system to provide the necessary information on demand.

Vision, in principle, has the capacity to provide the necessary information for a large number of complex and practical navigational operations. Ample evidence of this is provided by biological systems. There are a number of proposed applications of automation, including automatic guidance of ground vehicles, various "smart" weapons systems, construction or repair in hazardous environments, and exploration of planetary surfaces, which require interactive navigational abilities which cannot be provided by simple sensory systems of the type discussed above, and which, moreover, seem to be provided by visual means in biological systems. This is the practical motivation for the study of visual navigation.

The second, more philosophical motivation addresses the manner in which the study of vision can itself be most profitably approached. One method, which might be called the top down approach, starts by formulating a general

theory independent of the use to which the visually obtained information is to be put. Hopefully, specific applications will follow. Such theories have the advantage that they can stand on their own without being tied to any particular application. The corresponding disadvantage is that theoretical development can proceed almost indefinitely without necessarily generating any practically usable techniques. An example of such a situation is the mathematical development of shape from motion theories. A great deal of excellent theoretical work has been done in this area. Mathematical frameworks have been set up, theorems proven, algorithms developed, and literally hundreds of papers have been published. However, the theory has found little practical application because the algorithms tend to be extremely sensitive to inaccuracy in the input, and it has proven extremely difficult to obtain motion estimates from real image sequences which are sufficiently accurate to yield usable results. This is not to say that the effort was wasted. A lot has been learned in terms of understanding the information potentially available and the inherent limitations of the methods, and some of the techniques developed may eventually prove useful in certain situations. However, as a practical means of extracting precise three dimensional structural information about the environment, the technique has not lived up to its original claims.

An alternative method, which might be termed the bottom up approach, as already discussed, is to seek an understanding of visual processes in the context of specific problems by developing systems which actually perform certain practical tasks visually. Hopefully, commonalities observed among several such systems will allow the eventual formulation of a more general theory. Such a method is advocated by Brooks who suggests that artificial intelligence should be developed by building robots with increasingly sophisticated abilities. This approach lacks the intellectual appeal of a general theory, and has been criticized on the grounds that it will produce results of too narrow a scope, and without adequate theoretical foundation. On the other hand, if it produces anything at all, it is guaranteed to produce results which can actually be applied. Moreover, there is no reason why a solution to a specific problem cannot have a solid theoretical foundation within the domain of its applicability. Although the ability to design such working mechanisms for specific tasks does not necessarily demonstrate complete understanding of the problems of vision in general, no computational theory which does not provide such a capability should be considered adequate. In particular, theories of vision which are based on the results of some "lower level" processing which has not been demonstrated, or which utilize unrealizable assumptions about the mathematical nature of the environment, should be viewed as partial solutions at best, with the realization that the hardest problems probably remain to be solved.

The preceding discussion can be viewed in the light of the Marr paradigm. If this paradigm is accepted, then possession of an adequate model for any process implies the ability to build (given sufficient resources) a working mechanism.

Ideally then, a computational theory of vision in a given domain should eventually lead to a demonstrated ability to perform visual tasks using real data from that domain. There is, however, a disconcerting lack of visual systems which perform well in real-world environments, particularly when compared to the amount of high-powered mathematical theory which is published on the subject. There seem to be several reasons for this. First, it is very difficult to get a system working on real data at all, particularly one which is obviously impressive. The area of mathematical theory has historically proven more fruitful in terms of generating new and interesting results; furthermore, a mathematical theory may still be of interest mathematically even if it proves to be inapplicable in practice, whereas an actual system which does not work is of very little interest to anyone. Second, there has been a perception that practical results will eventually flow from a successful theory rather than vice versa. This perception probably has more to do with the lack of any practical systems to work with than with philosophical conviction, since historically, empirical engineering applications or unexplained observations have preceded theoretical developments at least as frequently as the reverse. If there were suddenly to appear a number of vision systems working robustly in different real-word domains, it is practically certain that an equal number of theories explaining their commonality would soon appear.

There is also a third reason that may explain the dearth of examples of working vision systems, which is that the generally accepted goals for such systems may be misplaced, or at least over-ambitious. There are two commonly held touchstones for practical vision systems. These are, one, a system which generates an accurate three dimensional description of objects in the world from one or more input images, and, two, a system which can recognize classes of objects (such as people or trees) in a complex scene with something approaching human reliability. A large proportion of the published papers on computer vision address, at least implicitly, one of these two goals. These are very high level objectives. If both were achieved, automatic systems would have many of the capabilities of the human visual system. Given the lack of success in developing systems which realize either of these goals in any robust manner, it would seem reasonable to consider simpler problems. There are applications which do not require the full realization of either capability, yet which are both non-trivial, and potentially useful. To take two biological examples, the housefly can maneuver visually in three dimensions in a complex environment without striking obstacles, and a number of bees and wasps can recognize and return to a particular location in such an environment from an arbitrary nearby starting point, an ability referred to as homing. Human beings can perform these tasks, but obviously they can be performed with far less computational equipment than humans posses. It would seem reasonable to consider some of these more restricted, specific problems with a view towards producing examples of visual systems which can actually be demonstrated to work well on real-world data. Such restricted problems appear to provide the best hope of producing a machine vision system fulfilling the criteria of the Marr paradigm.

Visual navigation represents an ideal example of a specific problem which could be profitably studied. First, it is a practical problem as has been mentioned already. Second, navigational abilities form a natural hierarchy beginning with simple abilities such as orientation and obstacle avoidance, and extending to more complex ones such as pursuit of a target and moving using specific knowledge about an environment. Lower levels can operate as independent systems, though a higher level module may depend on, or interact with, a lower one. For instance, a module plotting an interception course with a moving target might depend on an orientation module to keep the system stable with respect to the environment while the target trajectory is determined.

The hierarchy of levels provides an approach to producing practical visual systems of increasing complexity. Finally, providing information for navigation is one of the primary, and at the same time, one of the most basic tasks of natural vision systems. Almost all mobile organisms which are photosensitive to any degree use that sensitivity to guide their movement. If any heed is to be paid to the role of vision in biological systems, there is an important connection between vision and navigation.

### 4.1.1. Qualitative methods

Work in computer vision can be divided into two broad classes which are sometimes termed the reconstructive school and the recognition school, as already emphasized. Both schools have some modest successes to their credit, but the most notable result has been the discovery, if it can be called that, of the almost incredible computational difficulty of the visual abilities which human beings take for granted. This point has been driven home by the repeated failure of theoretically plausible models to yield usable systems. This is most notable in the reconstructive school where the criteria for success or failure are fairly clearly defined. Excluding structured light analysis, the most successful of the reconstruction methods are various stereo techniques, some of which work reasonably well on real images from certain restricted domains. There is a large amount of theoretical work on shape from motion in various forms, but so far, this has not resulted in practical methods for accurate reconstruction from real images. A number of other techniques have similarly robust theoretical development, and even more dismal practical performance. Success or failure in the recognition school is somewhat more difficult to judge since there is no direct quantitative connection. The most successful application of recognition techniques have been in two-dimensional binary applications such as optical character recognition and various industrial inspection tasks. There has been some success in highly domain specific scene understanding problems, but such systems tend to be quite brittle.

Visual navigation has generally been considered to be a subproblem of the reconstructive school of vision. The connection is a natural one, since navigation involves shape and distance relationships between the system and the environment which can be expressed in terms of the quantitative idiom of the reconstructive school. This perception has tended to discourage explicit research on visual navigation since it is considered as a special case of an important general problem. It has also tended to obscure the fact that many of the operations necessary for navigation can be expressed in qualitative terms which are more aptly described in terms of the recognition idiom. Consider, for example, the problem of kinematic stabilization. It is not necessary to know exactly how the system is moving with respect to the environment, but only whether it is rotating or translating at all, and if so, in what direction forces must be applied to reduce the motion. In the case of obstacle avoidance, the most directly relevant information is not the exact distance in inches from the observer to each point in the environment, but whether the system is on a collision course with a nearby obstacle, and, if so, in which direction it should move to avoid the danger of a crash. In the case of homing, it is not necessary to know, for example, the entire B-spline shortest path to the goal, but just a direction of movement that will bring the system closer to the goal.

The common factor in the above examples is that they do not require precise quantitative information, and that in each case, the information necessary to perform the task can be represented in a space having only a few degrees of freedom. This is exactly the kind of qualitative information which can be provided by pattern recognition systems. The important point is that, although solving any of the problems of navigation using quantitative techniques requires essentially a complete solution of the reconstruction problem, many of the same problems can be solved by qualitative techniques using restricted forms of pattern recognition. As we shall see, the pattern recognition abilities required for some of the basic navigational skills appear to be particularly simple. The hope is that, by applying qualitative methods to visual navigation, practical systems can be developed which do not depend on a full solution to the quantitative reconstruction problem.

### 4.1.2. Basic navigational abilities

It was noted above that navigational abilities fall into a natural hierarchy. Here we consider three of the most basic of these abilities, namely, passive navigation, obstacle avoidance, and simple visual homing. Passive navigation is the ability of a system to determine its own motion with respect to the environment. Obstacle avoidance implies the capacity to move about in an environment containing physical objects without striking them. Homing refers to the ability to reach a special point in the environment from an arbitrary starting location. These problems can all be approached by qualitative, pattern-recognition techniques. Moreover, together they constitute a solid set of elementary navigational tools for practical application. It is interesting to note that these navigational abilities can all be found in the simpler biological vision systems such as those possessed by flying insects. Perhaps this is a reason to hope that they may be easier to emulate than human visual skills. Below, we consider the problems individually, and propose qualitative methods of approaching them.

Passive navigation is a term used to describe the processes by which a system can determine its motion with respect to the environment. This is important for kinetic stabilization which, in its simplest form, requires a system to maintain a fixed position and attitude in space in the presence of perturbing influences. More generally, stabilization can refer to any conditions placed on the motion parameters; for instance, the system might be required to translate without rotation. The two abilities are interrelated because stabilization is generally achieved by bringing the motion parameters to certain specified values. The capacity for passive navigation is prerequisite for any other navigational ability. In order to guide the system, some idea of the present motion and some method of setting it to known values must be available. In present robot systems the necessary information is often explicitly available as a result of a built-in coordinate system. For an autonomously moving system, however, there must be an active sensing capacity. It is possible to obtain the required information mechanically as is done by the inertial guidance system in guided missiles. However, the task can also be performed by visual means. It is in this possibility that we are interested.

Passive navigation, as it was originally formulated, utilizes estimates of the projected motions of scene points to determine both the motion of the observer and the three-dimensional positions of the scene points. A number of varia-

tions have been explored in great detail. Ullman proposed that sets of corresponding points at discrete times could be used, and showed that at least three views of four points were required. A large body of work has been published concerning the theoretical aspects of this method. A second method utilizes the two dimensional projection of the relative motion of the scene points. This vector field is variously referred to as the motion field or the optical flow. Various authors have described methods which utilize various derivatives of this field to obtain information about the motion of the observer and the shape of the environment. In general, these methods make certain assumptions about the available information, and derive sets of equations which are then solved for the motion parameters and structural information. The main problem with all of these methods is that the information on which the theories are based, correspondence in the first case, and projected motion in the second, is extremely difficult to obtain accurately from real image sequences. Moreover, the mathematical equations are typically unstable with respect to small amounts of error. This has prevented these methods from finding much practical application. There are some recent results which suggest that the shape from motion problem, at least in certain aspects, may be essentially ill conditioned.

The sensitivity to error in the quantitative methods mentioned above is primarily due to the fact that the optical flows due to very different motions of the observer can be quite similar locally. However, topological properties guarantee that such flows cannot be similar over the entire visual sphere. With this in mind, we have proposed a method of determining the motion parameters of the observer using the motion field over a 360 degree spherical field of view. The method is based on the observation that the spherical motion field for a observer translating without rotation contains a focus of expansion and a focus of contraction separated by 180 degrees, and is parallel to the geodesics connecting the two foci. It can be shown that no motion containing non-zero rotation can produce a motion field satisfying these conditions. Thus the presence or absence of a particular, simple pattern is sufficient to establish whether or not the observer is rotating [Nelson and Aloimonos, 1987].

The component of the motion field produced by rotational motion is purely geometric, and independent of the three dimensional structure of the environment. The effect of a specified rotational component can thus be removed from the motion field by a process of *derotation*. Thus, in theory, the rotational components of motion could be determined by calculating the derotated flow field for different combinations of rotational components until the pattern characteristic of pure translational motion appeared. Unfortunately, the search space is too large to make this practical. However, it can be shown that the effects of the three rotational parameters decouple along the three equators corresponding to the principal axes. The rotational parameters can be determined independently by searching for a rotation parameter value for which the derotated circular flow field along the corresponding equator can be partitioned into disjoint semicircles of clockwise and counterclockwise flow. This is also a pattern recognition problem, but the search space has been reduced from five dimensions to two. The original problem is thus decomposed into three independent problems which can be solved with a simple best match search.

The above technique is practical to implement, and can be theoretically shown to be insensitive to inaccuracy in the input.

There are two main points to be made here. The first is that by considering the full visual sphere, rather than a restricted field of view, the problem of determining the motion parameters can be solved much more robustly. The second is that, once cast in this form, the problem can be solved using a simple, qualitative algorithm, with no need to solve complex sets of equations. It is interesting to note in this connection that flying insects which rely on vision for navigation have fields of view which subtend almost 360 degrees.

Obstacle avoidance refers, simply, to the ability to utilize sensory information to maneuver in an environment containing physical objects without striking them. This can be considered a second-level ability. It requires some capacity for passive navigation, but little else, and could thus be considered the lowest level of active navigation. This task can be performed non-visually by range sensing methods, and it is generally proposed that the problem be solved visually with a similar algorithm utilizing depth data from a reconstructed scene. Visual obstacle avoidance has, consequently, received little individual attention. Nevertheless, it is an important basic ability, and one which seems to have a surprisingly simple and robust qualitative solution.

The method we propose uses the fact that the expansion of an approaching obstacle produces positive divergence in the optical flow, and the fact that divergence is invariant under rotational motion. Thus a divergence detector would respond to approaching obstacles regardless of the motion of the observer. Moreover, regional divergence, such as that associated with an approaching obstacle, is a persistent feature in the sense of being robust under perturbation of the input data. Our system makes use of a set of one-dimensional divergence-like measures called *directional divergences*. These are essentially the one-dimensional divergences of the projections of the motion field in various directions. They have the advantage both of being easier to compute from image sequences, and of conveying more information than the single divergence value. It can be shown that the directional divergence possesses the same qualities with respect to rotational invariance as the ordinary divergence.

The main theoretical difficulty in using divergence measurements for obstacle avoidance has been that motion towards the observer is not the only factor that can produce divergent flow. In particular, translation parallel to a tilted surface can result in positive or negative divergence, depending on the direction of the motion. If only the ordinary divergence is used, the combination of such effects can make the interpretation of a divergence value difficult. For directional divergence, however, it is possible to show the following. First, for an observer undergoing arbitrary rotational and translational motion, any object having a component of relative motion towards the observer will produce a positive directional divergence in some direction. This means that no matter how the observer is moving, a potential collision can always be detected on the basis of divergence measurements. Second, detection of divergence in the flow field always indicates an object which is nearby in a sense which can be well defined. Thus even if the detection of divergence in some portion of the image does not indicate an imminent collision, it is worth noting because it represents a nearby obstacle which

would become a collision hazard if the direction of motion were to change. A much stronger result can be stated if the observer is undergoing purely translational motion as could be achieved using the stabilization method mentioned above. In this case, an object is on a collision course with the sensor if and only if there is positive divergence at a point of zero flow (i.e., a focus of expansion).

The practical question is whether the directional divergences can be determined with enough reliability for the above approach to be usable. Because only inexact estimates of the divergence are required, it turns out that a relatively simple procedure provides sufficient information from real images. Briefly, a differential technique (a la Horn) was used to approximate the projection of the motion field parallel to the local gradient at each point in the image. The information for each direction was then used to estimate the corresponding directional derivative by considering the difference of the average projected motion in adjacent neighborhoods. The method depends on the existence of enough visual texture for the differential method to "grab". In practice, the texture in ordinary objects such as cinder blocks and tree bark proved sufficient.

Using the above technique, we have developed an obstacle avoidance system for a robot mounted camera which was able to navigate successfully between obstacles in a real-world environment [Nelson et al., 1988].

The third problem we wish to consider is that of visual homing. This refers to the ability to move to a special point in the environment from a more or less arbitrary starting location. The system might also be required to reach the point in a specified orientation. This special point will be referred to as the home point or the zero point. The home point is not assumed to have any special qualities aside from its specification. It could be any point in the environment. An example of this ability is the behavior displayed by certain wasps which are able, upon returning from a considerable distance, to locate an obscure nest on the basis of surrounding visual features. This ability is slightly more complex than the preceding ones, as it requires the system to be trained to respond to features specific to a particular environment. It can also be considered a third-level ability, since a practical homing system would probably incorporate stabilization and obstacle avoidance. This problem is more obviously in the domain of pattern recognition than the two preceding operations. There is no simple method of utilizing a three dimensional reconstruction for its solution. Current systems which perform homing do so on the basis of a beacon—a unique, unmistakable feature. The challenge is to find a way of training a system to home using only the features naturally present.

It is proposed that the problem be approached by implementing a direct association between visual patterns and patterns of motor control. This idea is based on certain concepts of associative memory which have seen a broad resurgence in the past few years. The aspect of this work that is attractive with respect to the problem of homing is the capability of some systems to "learn"[2] a set of associations on the basis of exposure to a training set in such a way that an input which is close, but not identical to one of the inputs in the training set produces an output which is similar to the corresponding output. The first of these attributes is desirable because it provides a method of programming the reflexes required for a particular environment by showing the system how to move from certain points. The second attribute is important because the pattern-motion association function is expected to be mostly continuous. This continuity arises from the requirement that the vector describing the visual scene from one viewpoint be similar to the vector generated for a nearby viewpoint, and from the fact that, in most cases, the motion necessary to head for home is similar for points close to each other. Occasional discontinuities may arise in situations where it is necessary to decide which direction to go around an obstacle, but these will be the exception rather than the rule. The ability to generate a meaningful output for an input which is similar to, but not identical to one in the training set is sometimes referred to as generalization. Since it is clearly impractical to include every possible viewpoint in the training set, the design of a homing system which utilizes a reasonable amount of storage will depend critically on this ability.

The simplest associative system possessing the above properties is the linear associative network described by Kohonen and others. These networks implement a general linear transformation between the input and the output vectors. Because the transformation is linear, certain limitations apply to the associations that can be stored. For instance, discontinuous transformations cannot be represented. More generally, similar input vectors cannot be easily made to map to highly dissimilar outputs without severely decreasing the storage capacity of the network. Practically, this is manifested by increasing error in the output vectors as the stored representations interfere with each other. Such interference is referred to as crosstalk, and it is minimized if the input vectors are orthogonal. On the other hand, these systems are relatively well understood, and the training procedures relatively efficient. Moreover, if the input vector space is large, and typical input vectors are sparse (i.e., most of the entries are zero), then problems with crosstalk can be reduced since separate inputs are unlikely to have many members in common.

A number of more complicated systems have been investigated; examples include Boltzman machines, non-linear multi-level networks, and networks of complex elements. Most of these systems, however, are not well understood, and significant questions remain open about the correctness or even the existence of the representations produced by such neural networks. Moreover, the few training algorithms which have been proposed for these systems are extremely slow, even for simple problems. However, preliminary investigations indicate that the linear associative network, with the addition of some simple non-linear pre- and post-processing stages, will be sufficient for encoding homing knowledge.

The difficult part of designing the system seems to lie not in the design of the associative network, but in the choice of the appropriate factors for use in the visual input vectors. It was mentioned above that in order for generalization to work, the mapping from system positional coordinates to input vectors must posses a degree of continuity. There is a balance that must be struck in this regard. In order for the system to be able to generalize, the features in the input vector must be detectable over some range of system positions. However, the more general a feature is, the less use it is in

[2]"Learning" here means adaptive estimation.

accurately specifying a particular movement. Yet the system must both be able to generalize in order to conserve storage, and be able to make fine course corrections near the goal. A possible solution to this dilemma is to provide the system with different associative memories for use in different domains. In the simplest example of such a scheme, the system would use the contents of one memory to make coarse adjustments while far from the goal, and then when it got close enough, switch over to another memory which utilized features at a smaller spatial scale to make fine course corrections.

This scheme can be modeled as a combination of a set of associative networks with a finite state machine. In the appropriate situation (for example, when the coarse associative network no longer indicates a strong correction), the machine switches states, and a new network is brought into play. Such a system has the potential to execute much more complicated actions than the simple homing described above. For instance, multiple goal locations could be stored, to be activated in the appropriate situations. Longer paths could also be executed by storing a series of subgoals which are activated in the appropriate sequence.

### 4.2. Object Recognition

In order to study object recognition we must develop its mathematics, i.e., a vocabulary for talking about it, and hopefully discover constraints. A major difficulty in object recognition is the fact that an object image on the viewer's sensor (camera, eyes) depends on the point of view, and there are infinitely many points of view. So, we consider the important problem of finding visible properties of an object that are invariant to the point of view. We have studied invariants of the general projective transformation, which includes both perspective and orthographic projections as special cases. Our literature search has identified classical theories for differential and algebraic invariants not used before in image understanding. As they stand, these theories are not directly applicable to vision. We have suggested extensions and adaptations of these methods to the needs of machine vision. A clear exposition of this topic can be found in [Weiss, 1988]. We are currently exploring statistical projective invariances.

Several studies demonstrate that object recognition is connected in an intrinsic way to matching, and matching is intractable. Thus, we need to develop approximation algorithms for solving such problems. With respect to matching, we have studied two stage matching procedures as applied to labelled graphs and other domains relevant to computer vision. We do not require that the match be exact but only that it satisfy a specified error criterion. It is shown that it is computationally more efficient to initially match a subgraph and check the rest of the graph only when this match succeeds. A probabilistic analysis of the expected cost of this procedure is given with the aim of determining the optimum subgraph size which minimizes this cost. The results are extended to graph matching with geometric constraints as well as to templates. A careful exposition of the probabilistic analysis of two stage matching is presented in [Sitaraman and Rosenfeld, 1987].

In parallel, research on discovering algorithms for solving approximately (intractable) interesting problems is being actively pursued in our laboratory.

### 4.3. Learning

Our efforts in the study of learning are centered around the discovery of the constraints that govern the problem under consideration. We are not trying to do learning from scratch, in which many parameters have to be learned. Recent research efforts have shown how boolean learning in neural networks is possible by learning the and-sets (conjunctions in a disjunctive normal form) for a given class of boolean formulas. Extensions introducing uncertainty (inaccurate feedback, noise, etc.) are under development.

## 5. CONCLUSIONS

We have summarized a large body of our current and recent research on understanding images. Our research is done both top-down and bottom-up in the Marr paradigm. We have enriched the Marr paradigm by considering the stability level, and we propose that robustness analysis should accompany any proposed algorithm. Finally, we should take into account all constraints and learn from nature through examples all undetermined parameters, thus avoiding ad hoc assumptions.

## REFERENCES


Aloimonos, J.(Y.) and Shulman, D., "Learning Shape Computations", *Proc. Image Understanding Workshop*, Feb. 1987, Los Angeles, CA.

Shulman, D. and Aloimonos, J.(Y.), "(Non)Rigid Motion Interpretation: A Regularized Approach", to appear in *Proc. Roy. Soc. Lond.*, **B**, 1988.

Nelson, R.C. and Aloimonos, J.(Y.), "Computing Motion Parameters from Spherical Flow Fields", to appear in *Biological Cybernetics*.

Kanatani, K. and Chou, T.C., "Tracing Finite Motions Without Correspondence", TR-1689, Center for Automation Research, University of Maryland, College Park, August 1986.

Chou, T.C. and Kanatani, K., "Recovering 3D Rigid Motion Without Correspondence", TR-1864, Center for Automation Research, University of Maryland, College Park, June 1987.

Aloimonos, J.(Y.) and Basu, A., "Combining Information in Low-Level Vision", TR-1947, Center for Automation Research, University of Maryland, College Park, January 1988.

Bajcsy, R., "Active Perception vs. Passive Perception", *Proc. Workshop on Computer Vision*, 1986.

Aloimonos, J.(Y.), Weiss, I. and Bandyopadhyay, A., "Active Vision", to appear in *Int'l. Journal of Computer Vision*, 1988.

Nelson, R.C. and Aloimonos, J.(Y.), "Using Flow Field Divergence for Obstacle Avoidance in Visual Navigation", TR-1914, Center for Automation Research, University of Maryland, College Park, September 1987.

Sitaraman, R. and Rosenfeld, A., "Probabilistic Analysis of Two Stage Matching", TR-1858, Center for Automation Research, University of Maryland, College Park, June 1987.

# An integrated approach to stereo matching, surface reconstruction and depth segmentation using consistent smoothness assumptions

Liang-Hua Chen and Terrence E. Boult
Columbia University Department of Computer Science
New York City, NY 10027 tboult@cs.columbia.edu

## Abstract

This paper presents a new algorithm for stereo matching which makes use of simultaneous matching, surface reconstruction, and segmentation of world surfaces. By integrating these three phases, which are traditionally temporally separated, the algorithm can make use of the current surface information to help disambiguate the potential matches.

After discussing the required mathematical background, the paper describes the integrated process of matching, reconstruction and segmentation. Unlike past attempts at integrating these processes, the presented algorithm uses a single smoothness criterion for both matching, reconstruction and segmentation. The segmentation part of the process is based on estimates of surface bending energy, and is significantly different from previous segmentation algorithms. Examples are presented showing results on both synthetic images and camera acquired images. The camera-based examples include both a traditional type scene with two objects, and a scene with transparent objects.

## 1 Introduction

Stereopsis is a technique for computing depth from two disparate images of a scene. This section discusses the background the problem which caused us to adopt our integrated approach. The processes of feature detection, matching and surface reconstruction and their inter-relationship are also discussed.

### 1.1 Background

As stereo vision is very important in many areas, the central task in stereo is to solve the correspondence problem, i.e., identify features in two images that are projections of the same entity in the three-dimension world. Once this is done, one can compute the distance to this entity. Ideally, one would like to find the correspondences of every individual pixel in both images of a stereo pair. However, it is obvious that the information content in the intensity value of a single pixel is too low for unambiguous matching. In practice, therefore, coherent collection of pixels are matched. These collections are determined and matched in two distinct ways ([Barnard and Fischler 82]):

(1) Area-based matching tries to match an area of pixels in one image to another image. A small window is chosen as the matching unit. A window in one image is matched with a range of windows in the other image using cross-correlation or similar measure of the similarity between two windows.

(2) Feature-based matching attempts to match some specific points, individual edge points, or linear edge segments which consists of chains aligned edge points. However, the feature matching necessarily leads to a sparse depth map and the rest of surface must be reconstructed by approximation.

Feature-based matching has been more effective in stereo (see [Medioni and Nevatia 85]), and the remainder of the paper will concentrate on an algorithm that uses this approach.

### 1.2 Motivation

Traditionally, there are a number of constraints that can be used to prune the possible search space for candidate matches. For example, many algorithms use the sign of zero crossing, the "orientation" of the feature, epipolar geometry, etc.. These constraints are heuristically-based on assumptions about the imaging system and feature detectors. Yet, in general, these constraints are insufficient to remove the matching ambiguity for all features, and systems must employ more potent assumptions.

Among the most common classes of powerful assumptions is the imposition of a smoothness constraint. Even before the advent of computer vision, it was noted in [Gibson 50] that depth usually varies "smoothly" across surfaces. Thus, the disparity values derived from matching should also vary "smoothly".* This smoothness constraint can then be used to further constrain the matching process, and thus resolve most of the ambiguities. While this is an important observation, it leaves the meaning of "smoothly" up to the reader.

In vision research there have been many stereopsis models proposed with different "smoothly" varying disparities, such as, the continuity constraint of [Marr and Poggio 79], figural continuity in [Mayhew and Frisby 81] (see also [Kim and Bovik 86]), the disparity gradient limit of [Koenderink and vanDoorn 76], (see also [Pollard, Mayhew and Frisby 85]), the analytic disparity fields of [Eastman and Waxman 85], and the local planar/quadric patches

*Note that this constraint does not hold at the boundary of three dimensional objects and therefore the disparity along projections of such discontinuities need not be smooth.

of [Hoff and Ahuja 87]. All the constraints are intended to enforce a model of surface smoothness. However, they only partially capture the desired model. There are several problems in the above models:

1. It is difficult to translate surface smoothness constraints into disparity smoothness constraints. Depth is a nonlinear function of camera geometry, pixel position, and disparity. Therefore, smoothness assumptions are different in disparity space and real world. (Most of the above references model smoothness in disparity space). While it may be possible to define a realistic smoothness assumption in disparity space, it seems more likely to be able to do so for world surfaces.

2. Obviously, the matching process provides constrains for surface reconstruction. One interpretation of the smoothness constraint is to impose conditions on the matching phase such that the resulting reconstructed surface is generally smooth.. Traditionally, matching and surface reconstruction are two separate and time-sequential processes. Thus, matching could not make use of information from the reconstruction stage.

3. There may be multiple surfaces in the image. An edge segment may crosses different surfaces (e.g. when the contrast between the boundary of surfaces is not strong enough), and the disparity will not vary smoothly along the edge segment. Thus, algorithms that trv to use a disparity smoothness over a window to locate the correct matcher will fail if the window cross several surfaces. This implies that surface segmentation must be incorporated into the matching process.

As to the surface segmentation problem, the most common approach is to determine the "discontinuity boundaries" in surface depth, surface orientation and/or surface curvature. This approach usually requires some reconstruction of the surface, and this presents numerous problems. First, in order to correctly reconstruct a surface, knowledge of the data segmentation is generally required. This results in a difficult chicken-and-egg problem. To make matters worse, the quality of the reconstruction in the neighborhood of an unmarked (i.e. as yet undetected) discontinuity is generally poor. Thus the localization of the discontinuity of iterative reconstruct/segment approaches, see e.g. [Terzopoulos 84] or [Hoff and Ahuja 87], will be questionable. Furthermore, any boundary-based segmentation approach will require considerable post processing to handle extended multiply connected objects (say behind a picket fence) and may never be able to handle transparent surfaces where locally there are only a few points on any one surface.

A final remark about traditional segmentation is related to the definition of "boundaries". It is well known that the perceived "boundaries" of surfaces in depth share many characteristics with subjective contours, see [Julesz 71], [Marr 81]. This suggests that a definition of "boundaries" in depth might be accomplished by some secondary processing which is shared with "boundary" detection from other visual modalities.

To ameliorate the above mentioned problems with boundary-based segmentation, this paper proposes that segmentation of 3D information should simply classify points as belonging to the same surface. The determination of boundaries will be relegated to some secondary process which will not be discussed here. [†]

In computer vision, as well as other domains, researchers have used minimal surface bending-energy as an assumption to aid in surface recovery, i.e., the acceptability of a "smooth surface" is inversely related to the energy (in term of a regularizing functional) of the surface, for example, see [Grimson 81], [Terzopoulos 84], [Wahba 84], [Franke 82], [Hoff and Ahuja 85], [Lee 85], [Choi and Kender 85], [Blake and Zisserman 86], [Boult 86], and [Lee and Pavlidis 87]. Thus, bending energy appears to be a natural choice for a "measure" determining if a group of points belong to the same surface. The bending energy of a surface $f$ is given by:

$$\left\{ \int\int_{\mathbf{R2}} \left( \left( \frac{\partial^2 f}{\partial x^2} \right)^2 + 2 \cdot \left( \frac{\partial^2 f}{\partial x \partial y} \right)^2 + \left( \frac{\partial^2 f}{\partial y^2} \right)^2 \right) dx\, dy \right\}^{\frac{1}{2}} \quad (1)$$

Of course, the above measure can only be the basis for a practical measure for segmentation. Other issues that must be addressed by a practicable measure include:

- Determination of the threshold for separation of a group or alternatively defining the tradeoff between the number of surfaces and sum of the energy of these surfaces to keep the system from segmenting the data into a large number of planar patches (which have zero energy). (The algorithm presented herein follows the first approach.)
- Careful determination of how to handle surface size or equivalently, the area over which the energy is measured.
- Relation of the energy to the number of data points,
- Determination of which point(s) in a group are the cause of a surface energy which is too high, i.e., the credit assignment problem.
- Relation of "depth" discontinuities and "orientation" discontinuities and how they effect the energy measure.

The authors acknowledge that there are numerous other measures of "surface smoothness" as might be implemented by parametric surface patches (e.g. [Allen 85]) or volumetric models (e.g. see [Rao, Nevatia and Medioni 87], [Boult and Gross 87], or [Bajcsy and Solina 87]. These approaches deserve careful consideration in future research efforts.

## 2 Mathematical Model and Tools

From the above discussion, what we need is a model of smooth world surfaces. By using such a model and some mathematical tools, we can simultaneously do matching, surface reconstruction and segmentation.

### 2.1 Definition of the Model of World Surfaces

The assumed model of world surface is intimately related to techniques for regularized surface reconstruction, see [Boult 86]. The

[†]Of course this view cannot be taken too far, there must be some limit to the number of possible "transparent" surfaces and some limit to the extent of any "disconnected" object which will be recognized as connected

class of surfaces used is defined as those functions (distributions) with their second derivative (in a distributional sense) in $H^{\frac{1}{2}}$, where $H^{\frac{1}{2}}$ is the Hilbert space of functions such that their tempered distributions $\nu$ in $\mathbb{R}^2$ have Fourier transform $\tilde{\nu}$ that satisfy

$$\int\int_{\mathbf{R}^2} \left( |\tau| \cdot |\tilde{\nu}(\tau)|^2 \, d\tau \right) \quad < +\infty.$$

This class of functions, referred to as $D^{-2}H^{\frac{1}{2}}$, is equipped with the second Sobolev semi-norm,

$$\| \cdot \|_{D^{-2}H^{\frac{1}{2}}} = \left\{ \int\int_{\mathbb{R}^2} \left( \left( \frac{\partial^2 f}{\partial x^2} \right)^2 + 2 \left( \frac{\partial^2 f}{\partial x \partial y} \right)^2 + \left( \frac{\partial^2 f}{\partial y^2} \right)^2 \right) dx \, dy \right\}^{\frac{1}{2}} \tag{2}$$

which makes it a semi-Hilbert space.

Intuitively these functions are smooth (almost everywhere) up to derivatives of order approximately 1.5, i.e., they are significantly smoother than membrane surfaces but are not as smooth as thin-plate splines. The motivation for this choice is this "intermediate" level of smoothing assumed, and is supported by the results of [Boult 87].

## 2.2 The Definition of Reproducing Kernel-Based Spline

An essential ingredient of the current algorithm, at least from the point of view of efficient serial implementation, is the use of the reproducing kernel-based spline reconstruction as described in [Boult 86]. This section introduces some aspects of that algorithm necessary for later discussions.

Among all functions in the above class, the surface reconstruction aspect of the segmentation algorithm is required to find the surface which minimizes

$$\lambda \cdot \sum_{i=1}^{n} \frac{(\sigma(x_i, y_i) - z_i)^2}{\delta_i} + \|\sigma\|_{D^{-2}H^{\frac{1}{2}}}$$

where the data $z_i$ at point $(x_i, y_i)$, $i = 1, \ldots, n$ is assumed to be on one surface. The *global smoothing parameter*, $\lambda$, should depend on the overall error in the initial data, and the factors $\delta_i$ allow for individual points to have greater "noise"; the factor $\lambda$ effects the overall tradeoff between surface smoothness (as measured by the norm $\| \cdot \|_{D^{-2}H^{\frac{1}{2}}}$) and the fidelity to the data points $z_i$ while the factors $\delta_i$ effects the contribution of a single data point so as not to penalize the surface as much (or to penalize it more, depending on the value of $\delta_i$) for not closely approximating the data at that point. Techniques for choosing these parameters have been discussed by other researchers, see [Bates and Wahba 82].

One solution to the above reconstruction problem is a reproducing kernel-based spline. It has already been shown, see [Meinguet 83], that for the above model of world surfaces, the appropriate reproducing kernel here is

$$K(x, y; u, v) = \gamma((x-u)^2 + (y-v)^2)^{\frac{3}{2}}$$

for a known constant $\gamma$

Given the above kernel, the spline which approximates the information

$$z = z_1, \ldots, z_k = \{f(x_1, y_1), \ldots, f(x_k, y_k), \quad i = 1, \ldots, k\}$$

can be expressed as:

$$\sigma_z = \sum_{i=1}^{k} \alpha_i K(x, y; x_i, y_i) + \beta_1 + \beta_2 x + \beta_3 y \tag{3}$$

where the constants $\alpha_i$ and $\beta_i$ are the solution to the system of linear equations:

$$\begin{array}{cccccccc} \mathcal{A}_{1,1} & \ldots & \mathcal{A}_{1,k} & \mathcal{B}_{1,1} & \mathcal{B}_{1,2} & \mathcal{B}_{1,3} & & z_1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ \mathcal{A}_{k,1} & \ldots & \mathcal{A}_{k,k} & \mathcal{B}_{k,1} & \mathcal{B}_{k,2} & \mathcal{B}_{k,3} & = & z_k \\ \mathcal{C}_{1,1} & \ldots & \mathcal{C}_{1,k} & \mathcal{D}_{1,1} & \mathcal{D}_{1,2} & \mathcal{D}_{1,3} & & 0 \\ \mathcal{C}_{2,1} & \ldots & \mathcal{C}_{2,k} & \mathcal{D}_{2,1} & \mathcal{D}_{2,2} & \mathcal{D}_{2,3} & & 0 \\ \mathcal{C}_{3,1} & \ldots & \mathcal{C}_{3,k} & \mathcal{D}_{3,1} & \mathcal{D}_{3,2} & \mathcal{D}_{3,3} & & 0 \end{array}$$

where (4)

$$\begin{array}{c} \mathcal{A}_{i,i} = \frac{\alpha_i \lambda 8\pi}{\delta_i} \quad i = 1, \ldots, k; \\ \mathcal{A}_{i,j} = \alpha_j (K(x_j, y_j; x_i, y_i)), \quad i, j = 1, \ldots, k, \; i \neq j; \\ \mathcal{B}_{i,j} = \mathcal{C}_{j,i} = \beta_j p_j(x_i, y_i) \quad i = 1, \ldots, k, \; j = 1, \ldots, 3; \\ \text{and} \qquad \mathcal{D}_{i,j} = 0 \; i = 1, \ldots, 3 \; j = 1, \ldots, 3; \end{array}$$

The important properties of the above solution to the surface reconstruction problem are:

1. The algorithm is efficient for very sparse data (anything more than 3 non-colinear points will do, and the fewer the number of points, the faster the surface can be computed).
2. The surface is defined by the solution to a linear system which depends only on the location of the data. If the solution to this system can be updated quickly, the surface can also be updated quickly.
3. The surface is given in a functional form, thus the evaluation of derivatives is trivial, and bounds on the energy of the surface can be computed analytically.
4. The surfaces are independent of the "boundaries" of discontinuities, and depend only on the data values. However, the actual surface will change if the number/value of data points on the boundary are changed.

## 2.3 Definition of the Energy Measure

The basic form of the energy measure is given by equation 2 except that the region of integration may be different than that expressed therein.

The energy of the surface will depend on the size of the region in $\mathbb{R}^2$ over which the energy norm is computed. The two most natural choices are $\mathbb{R}^2$ itself, and the convex hull of the data defining the "current" surface. Unfortunately, neither of these is appropriate. For the above class of functions, the integral over $\mathbb{R}^2$ is not necessarily finite. Although the energy norm over the convex hull of the data defining the "current" surface is obviously finite, this choice has two difficulties:

1. The convex hull would be continuously changing as new data points were added to a surface.
2. The use of a domain which ends near the data points will allow the addition of new points to actually lower the surface energy, thus the energy will no longer be monotonicly

increasing and segmentation could not proceed with a region growing method.

Because of the above difficulties with the "natural" choices for the domain of integration, the algorithm uses the following heuristic: given the starting basis, the algorithm computes the energy of the surface over a square region which is centered around the centroid of the data (including the points not yet considered) with the length of the side of the rectangle 100 times larger than the larger dimension of the rectangle bounding all data points.

## 2.4 Derivation of Bounds on Energy of Surface

Given the definition of the spline as in equation 3, one can symbolically compute bounds on the energy. To begin, the exact form of the energy integral is manipulated to explicitly expand the squaring operation and move the differentiation and integration inside the sum, to wit:

$$
\begin{aligned}
\|\sigma\|_{D^{-2}H^{\frac{1}{2}}} = & \\
& \left\{ \int_{X_l}^{X_u} \int_{Y_l}^{Y_u} \left( \left( \frac{\partial^2 \sum_{i=1}^{k} \alpha_i K(x,y;x_i,y_i)}{\partial x^2} \right)^2 \right. \right. \\
& \quad + 2 \cdot \left( \frac{\partial^2 \sum_{i=1}^{k} \alpha_i K(x,y;x_i,y_i)}{\partial x \partial y} \right)^2 \\
& \quad \left. \left. + \left( \frac{\partial^2 \sum_{i=1}^{k} \alpha_i K(x,y;x_i,y_i)}{\partial y^2} \right)^2 \right) dx\, dy \right\}^{\frac{1}{2}} \\
= & \left\{ \int_{X_l}^{X_u} \int_{Y_l}^{Y_u} \left( \left( \sum_{i=1}^{k} (\alpha_i \frac{\partial^2 K(x,y;x_i,y_i)}{\partial x^2}) \right)^2 \right. \right. \\
& \quad + 2 \cdot \left( \sum_{i=1}^{k} (\alpha_i \frac{\partial^2 K(x,y;x_i,y_i)}{\partial x\, \partial y}) \right)^2 \\
& \quad \left. \left. + \left( \sum_{i=1}^{k} (\alpha_i \frac{\partial^2 K(x,y;x_i,y_i)}{\partial y^2}) \right)^2 \right) dx\, dy \right\}^{\frac{1}{2}} \\
= & \left\{ \begin{array}{l} \sum_{i=1}^{k} \sum_{j=1}^{k} \left( \int_{X_l}^{X_u} \int_{Y_l}^{Y_u} (\alpha_i K_{xx}(x,y;x_i,y_i)) \right. \\ \qquad \left. \cdot (\alpha_j K_{xx}(x,y;x_j,y_j))\, dx\, dy \right) \\ + 2 \cdot \sum_{i=1}^{k} \sum_{j=1}^{k} \left( \int_{X_l}^{X_u} \int_{Y_l}^{Y_u} (\alpha_i K_{xy}(x,y;x_i,y_i)) \right. \\ \qquad \left. \cdot (\alpha_j K_{xy}(x,y;x_j,y_j))\, dx\, dy \right) \\ + \sum_{i=1}^{k} \sum_{j=1}^{k} \left( \int_{X_l}^{X_u} \int_{Y_l}^{Y_u} (\alpha_i K_{yy}(x,y;x_i,y_i)) \right. \\ \qquad \left. \cdot (\alpha_j K_{yy}(x,y;x_j,y_j))\, dx\, dy \right) \end{array} \right\}^{\frac{1}{2}}
\end{aligned}
\tag{5}
$$

While it would be most appropriate to symbolically integrate the terms in the last of the above equations, the authors (and MACSYMA) have been unable to obtain a solution. Fortunately, a symbolic solution can be obtained for bounds on the above equations. First note that if $V_i \geq 0, i = 1, \ldots, n$ then

$$
\begin{aligned}
& \sum_{i=1}^{k} \sum_{j=1}^{k} a_i \cdot a_j \cdot (min(V_i, V_j))^2 \\
& \quad < \sum_{i=1}^{k} \sum_{j=1}^{k} a_i \cdot a_j \cdot V_i \cdot V_j \\
& \quad < \sum_{i=1}^{k} \sum_{j=1}^{k} a_i \cdot a_j \cdot (max(V_i, V_j))^2.
\end{aligned}
\tag{6}
$$

In fact, the upper bound is trivially true even if some of the $V_i$'s are negative. Thus an upper bound on the energy integral above can be written in terms of similar to

$$
\sum_{i=1}^{k} \sum_{j=1}^{k} max \left[ \begin{array}{l} \left( \int_{X_l}^{X_u} \int_{Y_l}^{Y_u} (K_{xx}(x,y;x_i,y_i))^2, dx\, dy \right), \\ \left( \int_{X_l}^{X_u} \int_{Y_l}^{Y_u} (K_{xx}(x,y;x_j,y_j))^2 dx\, dy \right) \end{array} \right]
$$

While the general energy integral has not been computable in closed form, the above simpler integral is computable in closed form. In particular, one can derive:

$$
\begin{aligned}
& \int_{X_l}^{X_u} \int_{Y_l}^{Y_u} (K_{xx}(x,y;s,t))^2 dx\, dy = \\
& \left\{ \begin{array}{l} 9 \cdot \left[ tan^{-1}(\frac{X_u - s}{Y_u - t}) \cdot (\frac{X_u}{4} - s \cdot X_u^3 \right. \\ \qquad \left. + \frac{3}{2} s^2 \cdot X_u^2 - s^3 \cdot X_u) \right. \\ \quad + \; tan^{-1}(\frac{Y_u - t}{X_u - s}) \cdot (\frac{Y_u}{4} - t \cdot Y_u^3 \\ \qquad + \frac{3}{2} t^2 \cdot Y_u^2 - t^3 \cdot Y_u + \frac{1}{4}(t^4 - s^4)) \\ \quad + \; (Y_u - t) \cdot (X_u \cdot \frac{1}{12} \cdot (6 \cdot t \cdot Y_u - 3 \cdot (Y_u^2 - t^2)) \\ \qquad + \frac{1}{36} \cdot (3 \cdot X_u^3 + 9 \cdot (s \cdot X_u^2 - s^2 \cdot X_u))) \\ \quad + \; X_u \cdot (3 \cdot Y_u^3 + 9 \cdot (t \cdot Y_u^2 - t^2 \cdot Y_u)) \\ \quad + \; 3 \cdot Y_u \cdot (3 \cdot X_u^3 + 9 \cdot (s \cdot X_u^2 - s^2 \cdot X_u)) \left. \right] \end{array} \right. \\
& \begin{array}{l} -9 \cdot \left[ tan^{-1}(\frac{X_l - s}{Y_u - t}) \cdot (\frac{X_l}{4} - s \cdot X_l^3 \right. \\ \qquad \left. + \frac{3}{2} s^2 \cdot X_l^2 - s^3 \cdot X_l) \right. \\ \quad + \; tan^{-1}(\frac{Y_u - t}{X_l - s}) \cdot (\frac{Y_u}{4} - t \cdot Y_u^3 \\ \qquad + \frac{3}{2} t^2 \cdot Y_u^2 - t^3 \cdot Y_u + \frac{1}{4}(t^4 - s^4)) \\ \quad + \; (Y_u - t) \cdot (X_l \cdot \frac{1}{12} \cdot (6 \cdot t \cdot Y_u - 3 \cdot (Y_u^2 - t^2)) \\ \qquad + \frac{1}{36} \cdot (3 \cdot X_l^3 + 9 \cdot (s \cdot X_l^2 - s^2 \cdot X_l))) \\ \quad + \; X_l \cdot (3 \cdot Y_u^3 + 9 \cdot (t \cdot Y_u^2 - t^2 \cdot Y_u)) \\ \quad + \; 3 \cdot Y_u \cdot (3 \cdot X_l^3 + 9 \cdot (s \cdot X_l^2 - s^2 \cdot X_l)) \left. \right] \end{array} \\
& \begin{array}{l} +9 \cdot \left[ tan^{-1}(\frac{X_l - s}{Y_l - t}) \cdot (\frac{X_l}{4} - s \cdot X_l^3 \right. \\ \qquad \left. + \frac{3}{2} s^2 \cdot X_l^2 - s^3 \cdot X_l) \right. \\ \quad + \; tan^{-1}(\frac{Y_l - t}{X_l - s}) \cdot (\frac{Y_l}{4} - t \cdot Y_l^3 \\ \qquad + \frac{3}{2} t^2 \cdot Y_l^2 - t^3 \cdot Y_l + \frac{1}{4}(t^4 - s^4)) \\ \quad + \; (Y_l - t) \cdot (X_l \cdot \frac{1}{12} \cdot (6 \cdot t \cdot Y_l - 3 \cdot (Y_l^2 - t^2)) \\ \qquad + \frac{1}{36} \cdot (3 \cdot X_l^3 + 9 \cdot (s \cdot X_l^2 - s^2 \cdot X_l))) \\ \quad + \; X_l \cdot (3 \cdot Y_l^3 + 9 \cdot (t \cdot Y_l^2 - t^2 \cdot Y_l)) \\ \quad + \; 3 \cdot Y_l \cdot (3 \cdot X_l^3 + 9 \cdot (s \cdot X_l^2 - s^2 \cdot X_l)) \left. \right] \end{array}
\end{aligned}
\tag{7}
$$

$$
\left.
\begin{aligned}
-9 \cdot \Big[ & tan^{-1}\left(\frac{X_u - s}{Y_l - t}\right) \cdot \left(\frac{X_u}{4} - s \cdot X_u^3 \right. \\
& \left. + \tfrac{3}{2} s^2 \cdot X_u^2 - s^3 \cdot X_u\right) \\
+ \; & tan^{-1}\left(\frac{Y_l - t}{X_u - s}\right) \cdot \left(\frac{Y_l}{4} - t \cdot Y_l^3 \right. \\
& \left. + \tfrac{3}{2} t^2 \cdot Y_l^2 - t^3 \cdot Y_l + \tfrac{1}{4}(t^4 - s^4)\right) \\
+ \; & (Y_l - t) \cdot \left(X_u \cdot \tfrac{1}{12} \cdot (6 \cdot t \cdot Y_l - 3 \cdot (Y_l^2 - t^2)) \right. \\
& \left. + \tfrac{1}{36} \cdot (3 \cdot X_u^3 + 9 \cdot (s \cdot X_u^2 - s^2 \cdot X_u))\right) \\
+ \; & X_u \cdot (3 \cdot Y_l^3 + 9 \cdot (t \cdot Y_l^2 - t^2 \cdot Y_l)) \\
+ \; & 3 \cdot Y_l \cdot (3 \cdot X_u^3 + 9 \cdot (s \cdot X_u^2 - s^2 \cdot X_u))\Big]
\end{aligned}
\right\}
$$

Similar derivations exists for the two integrals

$$\int_{X_l}^{X_u} \int_{Y_l}^{Y_u} (K_{xy}(x, y; s, t))^2 dx\, dy$$

and

$$\int_{X_l}^{X_u} \int_{Y_l}^{Y_u} (K_{yy}(x, y; s, t))^2 dx\, dy$$

(The latter can, in fact, be obtained by a change of variables in equation 8.)

Combining equations 6 and evaluating the formulas as in 8 one can obtain closed form equations for the upper bound on the energy of a reproducing kernel-based spline. The lower bound is a bit more difficult. If the terms $K_{xx}(x, y; x_i, y_i)$, $K_{xy}(x, y; x_i, y_i)$, and $K_{yy}(x, y; x_i, y_i)$ were all nonnegative, then the bound from equation 6 would apply. Unfortunately, the terms may be negative.

For the segmentation process, it is considerably more convenient to use a single number (i.e. if energy $<$ threshold) rather than developing some technique to handle both upper and lower bounds. While the upper bound alone could be used, this seems to produce too conservative an estimate. Thus, throughout this paper, the phrase "energy" of a surface refers to the heuristic estimate given by the average of the upper bound and the "lower" bound from equation 6 divided by the number of points defining the surface. While this is theoretically a meaningless number, the results in later sections support this as a reasonable heuristic. When we derive a true lower bound, we believe that the same heuristic will be appropriate, only it will be more robust.

# 3 The Integration of Matching, Surface Reconstruction, and Segmentation

Matching, surface reconstruction and segmentation work "cooperatively" in this stereo algorithm. The first pass determines the potential matches for features, and the uniquely matching features determine initial depth data which is used for surface reconstruction (and segmentation). The current surface reconstruction provides the surface smoothness constraint which is used to disambiguate the remaining potential matching features, updates the surface reconstruction/segmentation as it goes.

The algorithm has five phases:

1. image acquisition and camera calibration,
2. feature detection,
3. determination of the potential matches, and the amount of ambiguity for each feature,
4. initial reconstruction and segmentation of surfaces,
5. disambiguation of the remaining ambiguous features with continual refinements of the segmented surface reconstructions.

Each of these phases is described in turn.

## 3.1 Image Acquisition and Camera Calibration

The stereo images were taken using a single camera at two different positions. Because of the rotation of the camera and lens distortion, it is difficult to have a horizontal epipolar line. However, we still can estimate the non-horizontal epipolar geometry (see the feature detection phase).[‡]

The aim of calibration is to calculate the perspective transformation matrix between 3-D world coordinates and image coordinates. The algorithm used by the system is based on a procedure in [Duda and Hart 73], see also [Ballard and Brown 82]. Given the measured 3-D world coordinates of a number of non-coplanar calibration points and the corresponding 2-D image coordinates, the coefficients in the perspective transformation matrix can be computed by least square solution. Given the perspective transformation matrix, the camera parameter can be calculated if needed (e.g., see [Ganapathy 84]), and the depth value of features can also be computed after the matching phase completes.

## 3.2 Feature Detection

The features used are an interest operator based on [Moravec 79], and the zero crossings of the Laplacian of the Gaussian (e.g. seee [Marr and Hildreth 80]). The reasons for using multiple features are:

1. Since the feature points of interest operator are very sparse, most of the points are uniquely matched. We can make use of the already matched pairs to estimate the non-horizontal epipolar geometry (mainly, the vertical disparity). This is needed to match the zero-crossings which cannot be easily distinguished vertically.
2. The zero crossing of the image provide a large number of features for matching algorithm, unfortunately the localization of these features is not highly accurate (especially if there are errors in vertical disparity). The features from the interest operator are not very dense, however, they provide very accurate localization of the features. By combining the two different types of features, we can avoid the problem that features of the stereo system are too sparse, have poor localization, or are sensitive to noise.

[‡] Although it is not difficult in principle to calculate non-horizontal epipolar geometry from camera parameters and imaging geometry, most stereo systems would rather use a parallel camera model to allow the use of the more efficient scan-line coherency constraint.

3. A final reason, possibly unique to our approach, is that we need a number of "unique matches" to build out initial surface reconstruction. The features from the interest operator generally produce a unique match, and thus supply numerous points for our initial surface reconstruction.

Additionally, to reduce error due to digitization and early processing, the zero crossing are thresholded based on the gradient magnitude. A quantitative argument about the threshold value was described in [Kim and Bovik 86]. If necessary, the output of interest operator also can be thresholded to ensure unique matching.

Since the number of points provided by two feature detectors are different, in building the resulting surface, each data point must give a weight. Otherwise, the zeroing crossing (with 1000-2000 points) would totally dominate the point's generated by interest operator (with 100-200 points).

### 3.3 Determination of Possible Matches and Feature Ambiguity

First, match the points produced by interest operators. Because these specific features are very sparse, the searching space can be expanded vertically, and the result will have little ambiguity. As mentioned above, after matching these features, the epipolar geometry of the image can be calculated. It is also assumed that some information about the experimental environment is known, e.g. the maximum and minimum depth, then by the perspective transformation matrix, the location and width of searching window (vergence window) can be estimated.

Secondly, for every non-horizontal zero crossing in the left image, a search is performed along the corresponding epipolar line. Assume the width of the searching window is L. A feature can match only those features in the window with similar features. For zero-crossings, "similar" is defined as having the same sign, and an orientation within $\pm 30°$ of the other feature.

Each possible matching feature within the window in the right image is considered for a given feature in the left image. If there is only one point, the match is considered unique, otherwise the number of potential matches ($< \frac{L}{2}$) characterizes the degree of ambiguity. Currently, the algorithm will reject any point which has more than $\frac{L}{2}$ possible matches.

### 3.4 Surfaces Reconstruction and Segmentation

After the determination of potential matches, those matches which were determined to be unique are converted into depth data. Surface(s) are reconstructed incrementally using reproducing kernel-based spline(s). Surface reconstruction and segmentation are two concurrent processes. The algorithm proceeds by building an initial approximation of a surface from the depth data.** Points are added to a surface as long as the addition does not cause the energy of said surface to exceed a certain threshold. When multiple surfaces exist, the different surfaces are tried, and the surface with the lowest energy will accept the point. If no surface can accept the point, a "new" surface is created and the point is added to that surface. This process is continued until all data points have been processed.

**For the current implementation, this is 4 data-points. The points are chosen as a local cluster, although this is not critical to the performance and may cause problems when transparent surfaces are considered.

### 3.5 Disambiguation of Ambiguous Features

After the initial surface(s) are "reconstructed" from the unique matches, they are used to disambiguate the remaining potential matches. The ambiguous matches are considered in increasing order of ambiguity, and within a given level of ambiguity, the reconstructed world-surface is incrementally updated by considering the matches from left to right and from top to bottom. For each ambiguous match, the algorithm uses the information from the calibration phase to compute the possible three dimensional coordinates of the "feature" for each of the possible matches. The potential match which corresponds to the three dimensional point "closest" to any of the existing surfaces is considered the correct match. The point is then added to the reconstructed world-surface, using the level of ambiguity to adjust the associated parameter $\delta_i$.

## 4 The Good points and the Bad Points of the Approach

This section critically reviews the algorithm described in this paper pointing out some of the major advantages +, major problems −, and some aspects which can be viewed as either a good or bad ±, depending on ones point of view.

+ The segmentation algorithm can handle transparent and occluded objects with few problems.

+ The segmentation process is based on surfaces having low bending energy, a heuristic which can be directly related to the physical process of surface formation.

+ The functional form of the reproducing kernel-based spline allows for direct estimation of the surface energy, thus making the segmentation process reasonably computationally efficient.

+ In the algorithm, multiple features are used. This reduces the need for the severe scan-line coherency constraint, while still allowing a large number of reliable features.

± In the camera calibration phase, only the perspective transformation matrix is calculated. This is flexible, allowing one to use a single camera to acquire stereo images. However, when base line information cannot be obtained correctly and there is a certain amount of vertical disparity, the depth value is best computed by a least square solution (i.e. not by the traditional triangulation techniques). Since we use least square method twice (the first time used is in calibration phase), one might expect the error to be large, however, the experimental results show the error of this procedure is still acceptable, partially because the reproducing kernel-based spline allows individual points to have greater "noise".

± The algorithm does not recover "boundaries" for the segmented data. This is advantageous because it allows for transparent and/or occluding surfaces, and because data is generally

sparse (and often noisier) near the boundary resulting in a poor boundary definition. This is a disadvantage because it requires a secondary process (possibly using ideas borrowed from work on subjective contour perception or Gestalt psychology) to determine the actual boundary. It is also is a disadvantage because depth discontinuities induce a region where no potential matches can exist. Because the algorithm does not develop boundaries, it cannot make use of this observation.

± The algorithm can easily be adapted to different measures of surface smoothness. This is advantageous because it allows for greater flexibility, but disadvantageous because determination of the most appropriate measure is difficult. The measure used in the experiments presented herein has proved to be a reasonable one.

± The algorithm uses reproducing kernel-based splines which are essentially a global surface reconstruction algorithm and provide for efficient serial implementation for sparse data (say $< 1000$ points per surface). If there are more points then the algorithm can be extended to use local reproducing kernel-based splines (loosely based on [Franke 82]), at the cost of making the surface definition localized to patches. The local method has been evaluated and performs reasonably well on large data sets but very poorly on sparse data (probably because some of the patches may have little or no data).

± Since the current algorithm segments the depth data by one pass, the order of processing of points will effect the resultant segmentation, especially when two surfaces come into direct contact and join in a rather smooth fashion (e.g. the wedge example above). This may actually be used to help in the segmentation process by processing the data in multiple orders and using any difference in data labeling to suggest a refined segmentation.

± The linear systems which define the splines are known to be moderately ill-conditioned, see [Boult 86]. This problem is exacerbated when the data used to define the splines is nearly linearly dependent. Unfortunately, because of the smoothness assumptions implied by the model, if two points are very close in $x, y$, (relative to the size of the area $x, y$ being considered), and have similar $z$ values, the information becomes more linearly dependent (if the Z values are different they will almost assuredly be segmented). Fortunately, and directly because it is almost linearly dependent, such information does not significantly effect the reconstructed surface. Thus, the algorithm can determine that such information is redundant and discard that information. Currently, this is done heuristically but future work will investigate the usefulness of such information in modifying the confidence of those points which are maintained by the system.

– The algorithm currently uses a heuristic approximation of the surface energy divided by the number of data points in the surface as a threshold for the segmentation. This is a hack, and future work must attempt to redress this issue. Luckily, this threshold for energy-based segmentation does not seem as sensitive as say thresholds for segmentation of an image based on intensity.

– The algorithm assumes one is interested in smooth surfaces and will most likely fail when this assumption is not satisfied. Unfortunately, the algorithm cannot even determine if the assumptions are satisfied (For example, consider a rough surface similar to a plane covered with a large number of small densely packed cones. If the data supplied to the algorithm are points on the background and the peak values of the cones, the algorithm is hopelessly doomed to predict two planar surfaces.)

– The algorithm is surfaced based, and cannot deal with data from multiple views of a volumetric object. Additionally, it will often fail if noise is such that a single $x, y$ location is assigned multiple data values (of the same type).

# 5 Experimental Result

One set of synthetic images and two set of real images are presented to illustrate the performance of the algorithm. Both are 512 by 512 with 8 bits of grey scale. The purpose of the synthetic images was to enable us to obtain some estimates of the error of the system. The other scene poses more realistic problems.

We first comment on the graphical display of surfaces. The reproducing-kernel splines (like almost any approximation algorithm) are not extremely good at extrapolation and display of the surface far from any data would be misleading. Since the reconstruction does not determine boundaries, there are no "clean" edges for display. Thus, the graphical display shows only the portion of the surface in the convex-hull of the data. The display of occluding or transparent surface is also difficult (with resorting to a ray-tracer) and thus, some of the surfaces are presented "floating" in space.

Figure 1 shows two planes synthetic image, the x, y value are in the range $[-.1, 2.0]$ with two synthetic camera locations at a distance of approximately 40 units. The equations of the underlying planes are $z = 0$ when $x < 1$, and $z = 2x$ when $x > 1$. The reconstructed surfaces can be seen in figure 2.

The system uses 2 synthetic images, first for calibration and then for the stereo processing. In the calibration phase the $z$ value for the center of each square is assumed known, and the image coordinates of each square are obtained by thresholding and computing the centroid.

After stereo processing, the estimated depth values were compared with the underlying plane equations, assuming the error was in the direction of left-camera position. The RMS of error is 0.049 with variance 0.045. The maximum value of the error was .111 on the plane $z = 0$ and .45 on $z = 2x$.

Figure 3 shows a stereo pair of images of a cup and a playing card. The range of depth values is 90 mm to 140 mm (measured in z direction), and the camera location was at about 500mm (left camera on the z axis). Figure 4 shows some of the output of two feature detectors. In this example, the algorithm found 264 unique matches and 923 ambiguous points which could be disambiguated. The remaining 230 points were rejected (declared unmatchable) either because they had too many potential matches or no potential matches with similar features. Figure 5 shows reconstructions after the algorithm has successfully performed matching, reconstruction and segmentation on the two surfaces in the scene.

The third example shows that the algorithm can work well

even with transparent (or extended, multiply connected) objects. In figure 6 the reader can see a few square labels on a wall behind a glass plate with triangular labels on it.

The range of depth value in the scene is -5 mm to 140 mm, and again the camera was at about 500mm. In this example, the algorithm found 155 unique matches and 786 ambiguous points. The other 204 points were rejected. Figure 8 shows the reconstructed surfaces.

Currently the algorithm requires about 20min (wall clock time) on a Vax750 when processing a 512 by 512 image.[††] This is an unacceptable time requirement for practical problems, and future work on optimization and possible parallel implementation will address this issue.

## 6 Conclusion

This paper has presented an integrated approach for stereo matching,visual surface reconstruction and segmentation. The algorithm uses a model of surface smoothness which can be based on physical properties, and which provide for a flexible choice of different smoothness measures. The segmentation algorithm does not determine boundaries between segmented surfaces which allows it to handle extended objects occluded by other objects and transparent objects. The algorithm has been successfully demonstrated one synthetic image and two real images examples, but further testing on more complex images is needed.

One possible disadvantage of the approach is that the overall reliability of the stereo system heavily depends on that of the first pass of matching. If the first pass of matching cannot be achieved reliably, the overall approach cannot succeed. If the unique (unambiguous) matching cannot always result in correct matching, then in the segmentation phase when the energy of a certain surface exceeds the threshold, there is no way to know whether a point should be on a different surface or if it is just a mismatched point (which should be discarded). Future work will investigate, on the assumption that the mismatched points are few, a way of identifying these points (probably using a local energy measure). In particular, we will examine the direct use of the energy measure to determine the best match from the potential match set.

The second fault is due to the fact that the segmentation algorithm cannot predict boundary contours. Thus, when the ambiguous point is close to the occluded boundary, it is difficult to decide which surface the point should be on, and cannot choose the correct match. Currently the algorithm chooses the potential match which corresponds to a 3 dimensional point "closest" to "any" of the existing surfaces. Future work will involve the development of an algorithm which can take the "clusters" of data points determined by the energy-based segmentation algorithm, and compute the subjective contours that determine their occluding boundaries.

Finally, the current approach depends on a global thresholding technique to realize the segmentation. Such a process is doomed to be troublesome unless a systematic determination of the threshold is possible. Future work will address this issue and will also investigate the use of adaptive thresholding (depending on the actual data) and the use of other properties, say rate of change of energy, as the means of realizing segmentation.

[††] Admittedly, like most code developed for research purposes there has been very little attempt to optimize the implementation of the algorithm. The major point of this research to date has been to show that the algorithms are reasonable and that they can be computed with moderate time complexity.

## Acknowledgments

This work was supported in part by Darpa contract #N00039-84-C-0165.

## References

[Allen 85] P. K. Allen. *Object Recognition Using Vision and Touch.* PhD thesis, University of Pennsylvania, Department of Computer Science., 1985.

[Bajcsy and Solina 87] R. Bajcsy and F. Solina. Three dimensional object representation revisted. In *Proceedings of the IEEE Computer Society International Conference on Computer Vision*, pages 231–240, IEEE, June 1987.

[Ballard and Brown 82] D.H. Ballard and C.M. Brown. *Computer Vision.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.

[Barnard and Fischler 82] S.T. Barnard and M.A. Fischler. Computational stereo. *Computer Surveys*, 14(4), December 1982.

[Bates and Wahba 82] D. Bates and G. Wahba. Computational methods for generalized cross-validation with large data sets. In C.T.H. Baker and G.F. Miller, editors, *Treatment of Integral Equations by Numerical Methods*, pages 283 296, Academic Press, New York, 1982.

[Blake and Zisserman 86] A. Blake and A. Zisserman. Invariant surface reconstruction using weak continuity constraints. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 62 68, IEEE, 1986.

[Boult 86] Terrance E. Boult. *Information Based Complexity in Non-Linear Equations and Computer Vision.* PhD thesis, Department of Computer Science, Columbia University, 1986.

[Boult 87] T.E. Boult. What is regular in regularization? In *Proceedings of the IEEE Computer Society International Conference on Computer Vision*, pages 457 462, IEEE, June 1987.

[Boult and Gross 87] T.E. Boult and A.D. Gross. Recovery of superquadrics from depth information. In *Proceedings of the AAAI Workshop on Spatial-Reasoning and Multisensor Integration*, AAAI, October 1987.

[Choi and Kender 85] D.J. Choi and J. R. Kender. Solving the depth interpolation problem with adaptive chebyshev acceleration method on a parallel computer. In *Proceedings of the DARPA Image Understanding Workshop*, pages 219 223, DARPA, 1985.

[Duda and Hart 73] R. O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* Wiley, NYC, NY, 1973.

[Eastman and Waxman 85] R.D. Eastman and A.M. Waxman. Disparity functionals and stereo vision. In *Proceedings of the DARPA Image Understanding Workshop*, pages 245–254, DARPA, 1985.

[Franke 82] R. Franke. Smooth interpolation of scattered data by local thin plate splines. *Comp. & Math. with Applications*, 8(4):273–281, 1982.

[Ganapathy 84] S. Ganapathy. Decomposition of transformation matrices for robot vision. In *International Conference on Robotics and Automation*, pages 130–139, 1984.

[Gibson 50] J.J. Gibson. *The Perception of the Visual World.* Houghton-Mifflin, Boston, 1950.

[Grimson 81] W. E. L. Grimson. *From Images to Surfaces: A Computational Study of the Human Visual System.* MIT Press, Cambridge, MA, 1981.

[Hoff and Ahuja 85] W. Hoff and N. Ahuja. Surfaces from stereo. In *Proceedings of the DARPA Image Understanding Workshop*, pages 98–106, DARPA, 1985.

[Hoff and Ahuja 87] W. Hoff and N. Ahuja. Extracting surfaces from stereo images: an integrated approach. In *Proceedings of the IEEE Computer Society International Conference on Computer Vision*, pages 284–294, IEEE, 1987.

[Julesz 71] B. Julesz. *Foundations of Cyclopean Perception.* University of Chiage Press, Chiago, IL, 1971.

[Kim and Bovik 86] N.H. Kim and A.C. Bovik. A solution to the stereo correspondence problem using disparity smoothness constraints. In *Proceegings of the IEEE conference on Systems, Man, and Cybernetics*, October 1986.

[Koenderink and vanDoorn 76] J. J. Koenderink and A. J. van Doorn. Geometry of binocular vision and a model for stereopsis. *Biological Cybernetics*, 21:29–35, 1976.

[Lee 85] D. Lee. *Contributions to Information-based Complexity, Image Understanding, and Logic Circuit Design.* PhD thesis, Department of Computer Science, Columbia University, 1985.

[Lee and Pavlidis 87] D. Lee and T. Pavlidis. One-dimensional regularization with discontinuities. In *Proceedings of the IEEE Computer Society International Conference on Computer Vision*, pages 572–577, IEEE, June 1987.

[Marr 81] D. Marr. *VISION.* Freeman, San Francisco, 1981.

[Marr and Hildreth 80] D. Marr and E. C. Hildreth. Theory of edge detection. *Proceeding Royal Society of London.*, B(207):187–217, 1980.

[Marr and Poggio 79] D. Marr and T. Poggio. A computational theory of human stereo vision. *Proceeding Royal Society of London.*, B(204):301–328, 1979.

[Mayhew and Frisby 81] J.E.W. Mayhew and J. P. Frisby. Psychological and computational studies towards a theory of human stereopsis. *Artificial Intelligence*, 17:349–385, 1981.

[Medioni and Nevatia 85] G. Medioni and R. Nevatia. Segment-based stereo matching. *Computer Vision, Graphics, and Image Processing*, 31:2–18, July 1985.

[Meinguet 83] J. Meinguet. Surface spline interpolation: basic theory and computational aspects. *Institut de Mathematique Pure dt Appliquee, Universite Catholique de Louvain*, 35, 1983.

[Moravec 79] H.P. Moravec. Visual mapping by a robot rover. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 598–600, August 1979.

[Pollard, Mayhew and Frisby 85] S.B Pollard, J.E.W. Mayhew, and J.P. Frisby. Pmf: a stereo correspondence algorithm using a disparity gradient limit. *Perception*, 14:449–470, 1985.

[Rao, Nevatia and Medioni 87] K. Rao, R. Nevatia, and G. Medioni. Issues in shape description and an approach for working with sparse data. In *Proceegings of the AAAI Workshop on Spatial Reasoning and Multi-sensor Fusion*, pages 168–177, St. Charles. IL, October 1987.

[Terzopoulos 84] D. Terzopoulos. *Multiresolution Computation of Visible-Surface Representations.* PhD thesis, MIT, 1984.

[Wahba 84] G. Wahba. Surface fitting with scattered noisy data on euclidean d-space and on the sphere. *Rocky Mountain Journal of Mathematics*, 14(1):281–299, 1984.

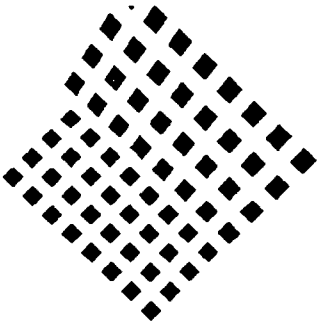


Figure 1: Synthetic image of two planes

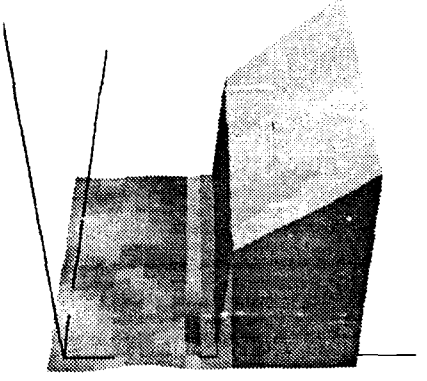


Figure 2: Reconstruction of two segmented planes

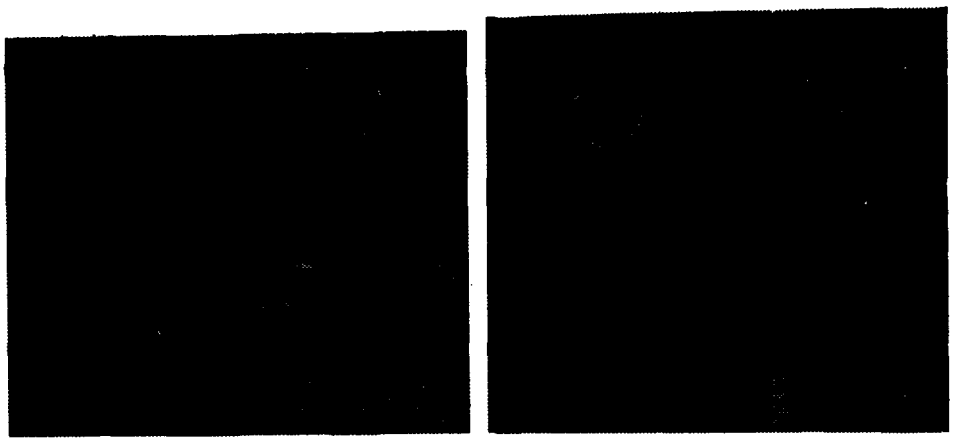


Figure 3: Left and right image of cup and poker card

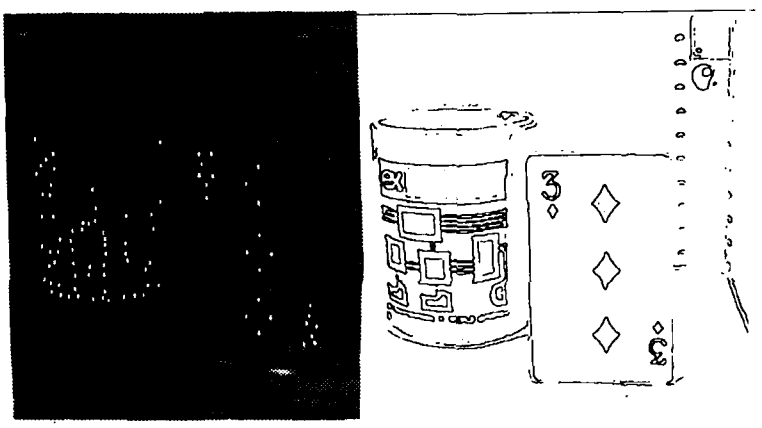


Figure 4: The left image is the output of interest operator, and the right image is the zero crossing of the Laplacian of the Gaussian.

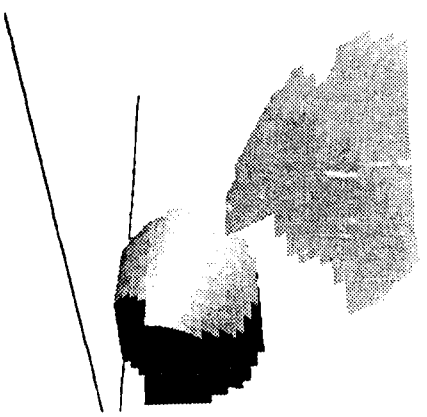


Figure 5: Reconstruction of two segmented surfaces

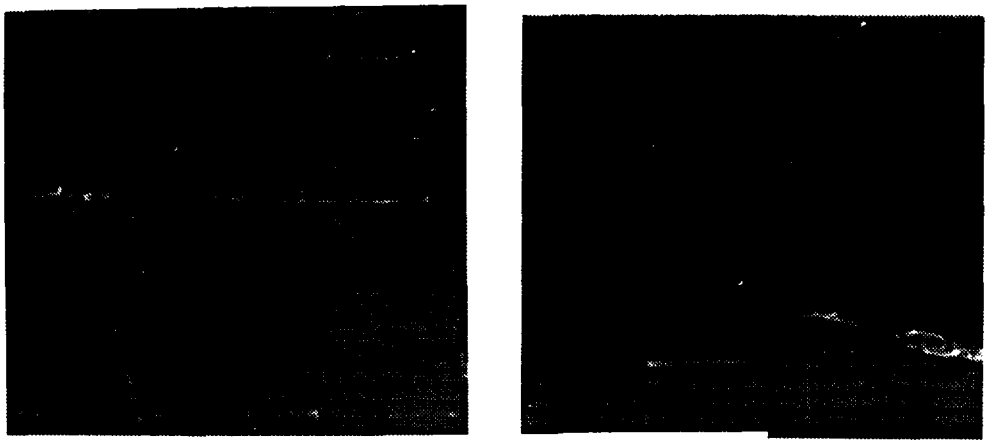


Figure 6: Left and right image of glass

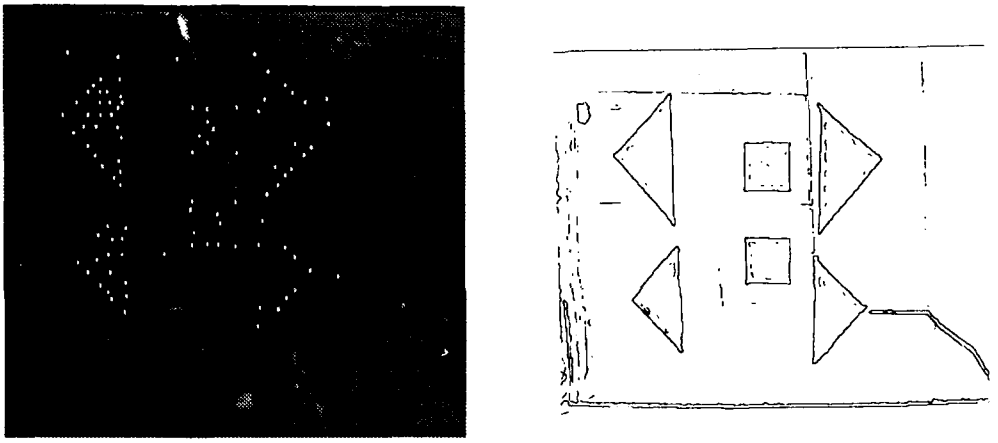


Figure 7: The left image is the output of interest operator, and the right image is the zero crossing of the Laplacian of the Gaussian.

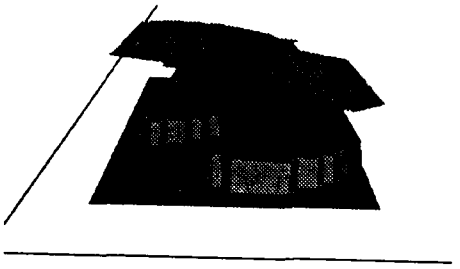


Figure 8: Reconstruction of two segmented surfaces

# THE MIT VISION MACHINE

T. Poggio, J. Little, E. Gamble, W. Gillett, D. Geiger
D. Weinshall, M. Villalba, N. Larson, T. Cass, H. Bülthoff
M. Drumheller, P. Oppenheimer, W. Yang, and A. Hurlbert

The Artificial Intelligence Laboratory
Massachusetts Institute of Technology

## ABSTRACT

*We describe the MIT Vision Machine, our goals and achievements to date. The Vision Machine is a computer system that attempts to integrate several vision cues to achieve high performance in unstructured environments for the tasks of recognition and navigation. It is also a test-bed for our theoretical progress in early vision algorithms, their parallel implementation and their integration. The Vision Machine consists of a movable two-camera Eye-Head system - the input device - and a 16K Connection Machine - our main computational engine. We have developed and implemented several parallel early vision algorithms which compute edge detection, stereo, motion, texture and surface color in close to real-time. The integration stage is based on the technique of coupled Markov Random Field models, and leads to a cartoon-like map of the discontinuities in the scene, with a partial labeling of the brightness edges in terms of their physical origin. We will interface the output of our integration stage with available recognition algorithms. We are also beginning to study analog and hybrid VLSI implementations of the Vision Machine main components.*

## 1. Introduction: The Project and Its Goals

Computer vision has developed algorithms for several early vision processes, such as edge detection, stereopsis, motion, texture, and color, which give separate cues as to the distance from the viewer of three dimensional surfaces, their shape, and their material properties. Biological vision systems, however, greatly outperform computer vision programs. It is increasingly clear that one of the keys to the reliability, flexibility and robustness of biological vision systems in unconstrained environments is their ability to integrate many different visual cues. For this reason we are developing a *Vision Machine System* to explore the issue of the integration of early vision modules. The system also serves the purpose of developing parallel vision algorithms since its main computational engine is a parallel supercomputer - the Connection Machine.

The idea behind the Vision Machine is that the main goal of the integration stage is to compute a map of the visible discontinuities in the scene, somewhat similar to a cartoon or a line-drawing. There are several reasons for this. Firstly, experience with existing model-based recognition algorithms suggest that the critical problem in this type of recognition is to obtain a reasonably good map of the scene in terms of features such as edges and corners. The map does not need to be perfect - human recognition works with noisy and occluded line drawings - and of course it cannot be perfect. But it should be significantly cleaner than the typical map provided by an edge detector. Secondly, discontinuities of surface properties are the most important locations in a scene. Thirdly, we have argued [Poggio, 1985] that discontinuities are ideal for integrating information from different visual cues.

It is also clear that there are several different approaches to the problem of how to integrate visual cues. Let us list some of the obvious possibilities:

1) There is no active integration of visual processes. Their individual outputs are "integrated" at the stage at which they are used, for example by a navigation system. This is the approach advocated by Brooks [1987]. While it makes sense for automatic, insect-like, visuo-motor tasks such as tracking a target or avoiding obstacles (e.g., the fly's visuo-motor system [Reichardt and Poggio, 1976]), it seems quite unlikely for visual perception in the wide sense.

2) The visual modules are so tightly coupled that it is impossible to consider visual modules as separate, even in a first order approximation. This view is unattractive on epistemological, engineering and psychophysical grounds.

3) The visual modules are coupled to each other and to the image data in a parallel fashion – each process represented as an array coupled to the arrays associated with the other processes. This point of view is in the tradition of Marr's $2\frac{1}{2}$-D sketch, and especially of the "intrinsic images" of Barrow and Tenenbaum [1978]. Our present scheme is of this type, and exploits the machinery of Markov Random Field (MRF) models.

4) Integration of different vision modalities is taking place in a task-dependent way at specific locations – not over the whole image – and when it is needed – therefore not at all times. This approach is suggested by psychophysical data on visual attention and by the idea of visual routines [Ullman, 1984; see also Hurlbert and Poggio, 1986; Mahoney, 1987].

We are presently exploring the third of these approaches. We believe that the last two approaches are compatible with each other. In particular, visual routines may operate on maps of discontinuities such as those delivered by the present Vision Machine, and therefore be located after a parallel, automatic integration stage. In real life, of course, it may be more a matter of coexistence. We believe, in fact, that a control structure based on specific knowledge about the properties of the various modules, the specific scene and the specific task will be needed in a later version of the Vision Machine to overview and control the MRF integration stage itself and its parameters. It is possible that the integration stage should be much more goal-directed that what our present methods (MRF based) allow. The main goal of our work is to find out whether this is true.

The Vision Machine project has a number of other goals. It provides a focus for developing parallel vision algorithms and for studying how to organize a real-time vision system on a massively parallel supercomputer. It attempts to change the usual paradigm of computer vision research over the past years: choose a specific problem, for example stereo, find an algorithm, and test it in isolation. The Vision Machine allows us to develop and test an algorithm in the context of the other modules and the requirements of the overall visual task – above all visual recognition. For this reason, the project is more than an experiment in integration and parallel processing: it is a laboratory for our theories and algorithms.

Finally, the goal of the Vision Machine project is no less than the ultimate goal of vision research: to build a vision system that achieves human-level performance.

## 2. The Vision Machine System

The overall organization of the system is shown in Figure 1. The image(s) are processed through independent algorithms or modules corresponding to different visual cues, in parallel. Edges are extracted using Canny's edge detector. Stereo computes disparity from the left and right images. The motion module estimates an approximation to the optical flow from pairs of images in a time sequence. The texture module computes texture attributes (such as density and orientation of textons [see Voorhees, 1987]). The color algorithm provides an estimate of the spectral albedo of the surfaces, independently of the *effective illumination*, that is, illumination gradients and shading effects, as suggested by Hurlbert and Poggio [see Poggio, 1985].

The measurements provided by the early vision modules are typically noisy and possibly sparse (for stereo and motion). They are smoothed and made dense by exploiting known constraints within each process (for instance, that disparity is smooth). This is a stage of *approximation* and *restoration* of data, performed by using a Markov Random Field model. Simultaneously, discontinuities are found in each cue. Prior knowledge of the behavior of discontinuities is exploited, for instance, the fact that they are continuous lines, not isolated points. Detection of discontinuities is aided by the information provided by brightness edges. Thus each cue - disparity, optical flow, texture, and color - is coupled to the edges in brightness.

The full scheme involves finding the various types of physical discontinuities in the surfaces – *depth discontinuities (extremal edges and blades), orientation discontinuities, specular edges, albedo edges (or marks), shadow edges* – and coupling them with each other and back to the discontinuities in the visual cues, as illustrated in Figure 1. So far we have implemented only the coupling of brightness edges to each of the cues provided by the early algorithm. As we will discuss later, the technique we used to approximate, to simultaneously detect discontinuities, and to couple the different processes, is based on MRF models. The output of the system is a set of labeled discontinuities of the surfaces around the viewer. In our implemented version of the system we find discontinuities in disparity, motion, texture, and color. These discontinuities, taken together, represent a "cartoon" of the original scene which can be used for recognition and navigation (along with, if

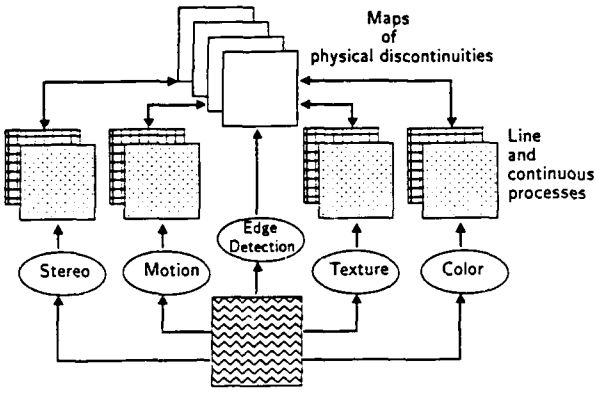



Figure 1: Block Diagram of the Vision Machine

needed, interpolated depth, motion, texture and color fields).

The plan of the paper is as follows. We will first review the present hardware of the Vision Machine: the Eye-Head system and the Connection Machine. We will then describe in some detail each of the early vision algorithms that are presently running and are part of the system. After this, the integration stage will be discussed. We will analyze some results and illustrate the merits and the pitfalls of our present system. The last chapter will discuss a real-time visual system and some ideas on how to put the system into VLSI circuits of analog and digital type.

## 3. Hardware

### 3.1. The Eye-Head System

Because of the variety of visual information processed by the Vision Machine, a general purpose image input device is required. Such a device is the Eye-Head system. Here we discuss its current and future configurations.

#### 3.1.1. The Present

The Eye-Head system (Figure 2a) consists of two CCD cameras ("eyes") mounted on a variable-attitude platform ("head"). The apparatus allows the cameras to be moved as a unit, analogous to head movement. It also allows the lines of sight of the cameras to be pointed independently, analogous to eye movement. Each camera is equipped with a motorized zoom lens ($F1.4$, focal length from 12.5 to 75mm), allowing control of the iris, focus, and focal length by the host computer (currently a Symbolics 3600 Lisp Machine). Other hardware allows for repeatable calibration of the entire apparatus.

Because of the size and weight of the motorized lenses, it would be impractical to achieve eye movement by pointing the camera/lens assemblies directly. Instead, each assembly is mounted rigidly on the head, with eye movement achieved indirectly. In front of each camera lens is a pair of front surface mirrors (Figure 2b), each of which can be pivoted by a galvanometer also mounted rigidly on the head. The mirrors are positioned to provide two degrees of freedom in aiming the cameras. At the expense of a more complicated imaging geometry, this allows for a simpler and faster control system for the eyes.

The head is attached to its mount via a spherical joint, allowing head rotation about two orthogonal axes (pan and tilt). Each axis is driven by a stepper motor coupled to the drive shaft through a harmonic drive. The latter provides a large gear ratio in conjunction with very little mechanical backlash. Under control of the stepper motors, the head can be panned 180 degrees from left to right, and tilted 90 degrees (from vertical-down to horizontal). Each of the stepper motors is provided with an optical shaft encoder for shaft position feedback (a closed-loop control scheme is employed for the stepper motors). The shaft encoders also provide an index pulse (one per revolution) which is used for joint calibration in conjunction with mechanical limit switches. The latter also protect the head from damage due to excessive travel.

The overall control system for the Eye-Head system is distributed over a micro-processor network (UNET) developed at the MIT AI Lab for the control of vision/robotics hardware. The UNET is a "multi-drop" network supporting up to 32 micros, under the control of a single host. The micros normally function as network slaves, with the host acting as the master. In this mode the micros only "speak when spoken to", responding to various network operations either by receiving information (command or otherwise) or by transmitting information (such as status or results). Associated with each micro on the UNET is a local 16-bit bus (UBUS), which is totally under the control of the micro. Peripheral devices such as motor drivers, galvanometer drivers, and pulse width modulators (PWMs), to name a few, can be interfaced at this level.

At present two micro-processors are installed on the Eye-Head UNET: one for the galvanometer and one for both the motorized lenses and stepper motors. The processors currently employed are based on the Intel 8051. Each of these micros has an assortment of UBUS

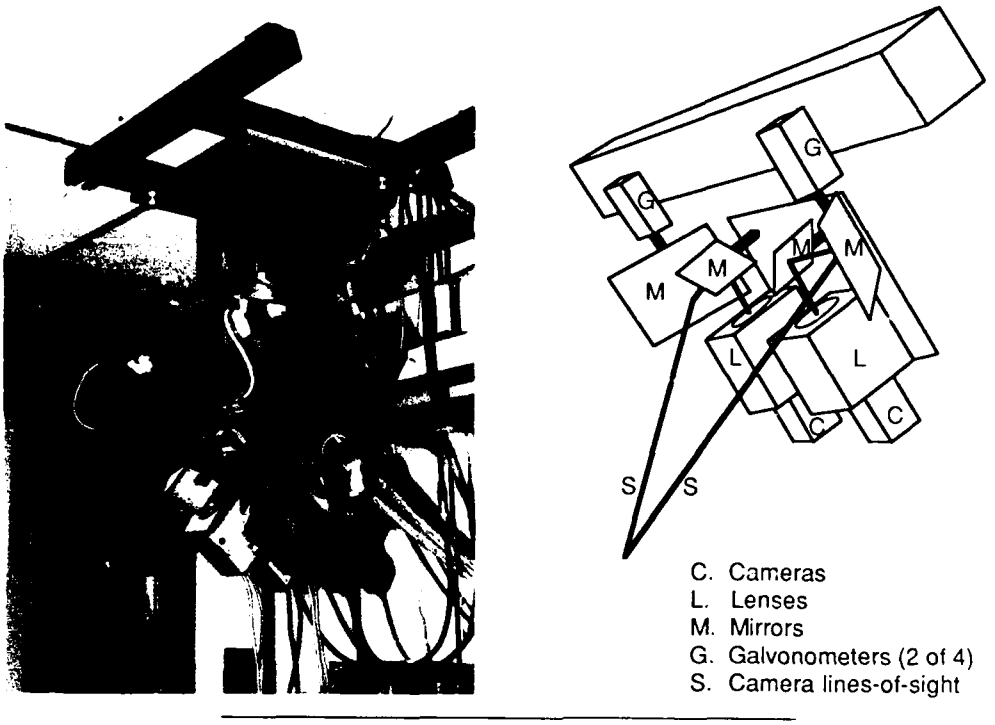



Figure 2: The Eye-Head System

peripherals under its control. By making these peripherals sufficiently powerful, each micro's control task can remain simple and manageable. Code for the micros, written in both assembly language and C, is facilitated by a Lisp-based debugging environment.

### 3.1.2. The Future

A single enhancement remains for the Eye-Head system. Currently, a Symbolics Lisp Machine acts as the host processor for the UNET. Soon an intermediate real-time processor will be placed between the Lisp Machine and the UNET, acting as master of the latter. The real-time processor (referred to as the DSP, being based on a Digital Signal Processor chip) will relieve the Lisp Machine of all the UNET protocol tasks, as well as various low-level, real-time control tasks for which the Lisp Machine is ill-suited. Among the tasks envisioned for the DSP is optimal position estimation of moving targets from motion data.

## 3.2. Our Computational Engine: The Connection Machine

The Connection Machine is a powerful fine-grained parallel machine which has proven useful for implementation of vision algorithms. In implementing these algorithms, several different models of using the Connection Machine have emerged, since the machine provides several different communication modes. The Connection Machine implementation of algorithms can take advantage of the underlying architecture of the machine in novel ways. We describe here several common, elementary operations which recur throughout the following discussion of parallel algorithms.

### 3.2.1. The Connection Machine

The CM-1 version of the Connection Machine [Hillis, 1985] is a parallel computing machine with between 16K and 64K processors, operating under a single instruction stream broadcast to all processors. It is a Single Instruction Multiple Data (SIMD) machine; all processors execute the same control stream. Each processor is a simple 1-bit processor, currently with 4K bits of memory. There are two modes of communication among the processors: first, they are connected by a mesh into a $128 \times 512$ grid network (the NEWS network, so-called because the connections are in the four cardinal directions), allowing rapid direct communication between neighboring processors, and second, the *router*, which allows messages to be sent from any

processor to any other processor in the machine. The processors in the Connection Machine can be envisioned as being the vertices of a 16-dimensional hypercube (in fact, it is a 12-dimensional hypercube; at each vertex of the hypercube resides a chip containing 16 processors). Each processor in the Connection Machine is identified by its hypercube address in the range 0...65535, imposing a linear order on the processors. This address denotes the destination of messages handled by the router. Messages pass along the edges of the hypercube from source processors to destination processors. The Connection Machine also has facilities for returning to the host machine the result of various operations on a field in all processors; it can return the global maximum, minimum, sum, logical AND, and logical OR of the field.

To allow the machine to manipulate data structures with more than 64K elements, the Connection Machine supports *virtual processors*. A single physical processor can operate as a set of multiple virtual processors by serializing operations in time, and partitioning the memory of each processor. This is otherwise invisible to the user. Connection Machine programs utilize Common Lisp syntax, in a language called *Lisp, and are manipulated in the same fashion as Lisp programs.

### 3.2.2. Powerful Primitive Operations

Many vision problems must be solved by a combination of communication modes on the Connection Machine. The design of these algorithms takes advantage of the underlying architecture of the machine in novel ways. There are several common, elementary operations used in this discussion of parallel algorithms: routing operations, scanning and distance doubling.

*Routing*

Memory in the Connection Machine is associated with processors. Local memory can be accessed rapidly. Memory of processors nearby in the NEWS network can be accessed by passing it through the processors on the path between the source and the destination. At present, NEWS accesses in the machine are made in the same direction for all processors. The *router* on the Connection Machine provides parallel reads and writes among processor memory at arbitrary distances and with arbitrary patterns. It uses a packet-switched message routing scheme to direct messages along the hypercube connections to their destinations. This powerful communication mode can be used to reconfigure completely, in one parallel write operation taking one router cycle, a field of information in the machine. The Connection Machine supplies instructions so that many processors can read from the same location or write to the same location, but since these memory references can cause significant delay, we will usually only consider exclusive read, exclusive write instructions. We will usually not allow more than one processor to access the memory of another processor at one time. The Connection Machine can combine messages at a destination by various operations, such as logical AND, inclusive OR, summation, and maximum or minimum.

```
processor-number = [0  1  2  3  4  5  6  7]
A                = [5  1  3  4  3  9  2  6]
Plus-Scan(A)     = [5  6  9 13 16 25 27 33]
Max-Scan(A)      = [5  5  5  5  5  9  9  9]
```

Figure 3: Examples of *Plus-Scan* and *Max-Scan*.

*Scanning*

The *scan* operations [Blelloch, 1987] can be used to simplify and speed up many algorithms. They directly take advantage of the hypercube connections underlying the router, and can be used to distribute values among the processors and to aggregate values using associative operators. Formally, the *scan* operation takes a binary associative operator $\oplus$, with identity 0, and an ordered set $[a_0, a_1, \ldots, a_{n-1}]$, and returns the set $[a_0, (a_0 \oplus a_1), \ldots, (a_0 \oplus a_1 \oplus \ldots \oplus a_{n-1})]$. This operation is sometimes referred to as the *data independent prefix operation* [Kruskal et.al., 1985]. Binary associative operators include *minimum*, *maximum*, *and plus*. Figure 3 shows scans using *maximum* and *plus*.

The four scan operations *plus-scan*, *max-scan*, *min-scan* and *copy-scan* are implemented in microcode and take about the same amount of time as a routing cycle. The *copy-scan* operation takes a value at the first processor and distributes it to the other processors. These scans operations can take *segment bits* that divide the processor ordering into segments. The beginning of each segment is marked by a processor whose segment bit is set, and the scan operations start over again at the beginning of each segment (see Figure 4).

The *scan* operations also work using the NEWS addressing scheme, termed *grid-scans*. These compute the sum, and find the maximum, copy, or number values along rows or columns of the NEWS grid quickly.

For example, *grid-scans* can be used to find for each pixel the sum of a square region with width $2m+1$ centered at the pixel. This sum is computed using the following steps. First, a *plus-scan* accumulates partial sums for all pixels along the rows. Each pixel then gets the result of the scan from the processor $m$ in front of it and $m$ behind it; the difference of these two values represents the sum, for each pixel, of its neighborhood

```
processor-number   = [0  1  2  3  4  5  6  7]
A                  = [5  1  3  4  3  9  2  6]
SB (segment bit)   = [1  0  1  0  0  0  1  0]
Max-Scan(A, SB)    = [5  5  3  4  4  9  2  6]
Copy-Scan(A, SB)   = [5  5  3  3  3  3  2  2]
Plus-Scan(A, SB)   = [0  5  6  3  7 10 19  2]
Min-Scan(A, SB)    = [MX 5  1  3  3  3  3  2]
```

Figure 4: Examples of Segmented *Scan* Operations.

along the row. We now execute the same calculation on the columns, resulting in the sum, for each pixel, of the elements in its square. The whole process only requires a few *scans* and routing operations, and runs in time independent of the size of $m$. The summation operations are generally useful to accumulate local support in many of our algorithms, such as stereo and motion.

*Distance Doubling*

Another important primitive operation is *distance doubling* [Wyllie, 1979; Lim, 1986], which can be used to compute the effect of any binary, associative operation, as in *scan*, on processors linked in a list or a ring. For example, using *max*, *doubling* can find the extremum of a field contained in the processors. Using message-passing on the router, *doubling* can propagate the extreme value to all processors in the ring of $N$ processors in $O(\log N)$ steps. Each step involves two *send* operations. Typically, the value to be maximized is chosen to be the hypercube-address. At termination, each processor in the ring knows the label of the maximum processor in the ring, hereafter termed the *principal processor*. This labels all connected processors uniquely and nominates a processor as the representative for the entire set of connected processors. At the same time, the distance from the *principal* can be computed in each processor. Figure 4 shows the propagation of values in a ring of eight processors. Each processor initially, at step 0, has the address of the next processor in the ring, and a value which is to be maximized. At the termination of the $i^{th}$ step, a processor knows the addresses of processors $2^i+1$ away and the maximum of all values within $2^{i-1}$ processors away. In the example, the maximum value has been propagated to all 8 processors in $\log 8 = 3$ steps.

# 4. Early Vision Algorithms and their Parallel Implementation

## 4.1. Edge Detection

Edge detection is a key first step in correctly identifying physical changes. The apparently simple problem of measuring sharp brightness changes in the image has proven to be difficult. It is now clear that edge detection should be intended not simply as finding "edges" in the images, an ill-defined concept in general, but as measuring appropriate derivatives of the brightness data. This involves the task-dependent use of different two-dimensional derivatives. In many cases, it is appropriate to mark locations corresponding to appropriate critical points of the derivative such as maxima or zeroes. In some cases, later algorithms based on these binary features – presence or absence of edges – may be equivalent, or very similar, to algorithms that directly use the continuous value of the derivatives. A case in point is provided by our stereo and motion algorithms, to be described later. As a consequence, one should not always make a sharp distinction between edge-based and intensity based algorithms: the distinction is more blurred and in some cases, it is almost a matter of implementation.

In our current implementation of the Vision Machine, we are using two different kinds of edges. The first consists of zero-crossings in the Laplacian of the image filtered through an appropriate Gaussian. The second consists of the edges found by Canny's edge detector. Zero-crossings can be used by our stereo and motion algorithms (though we have mainly used Canny's edges at fine resolution). Canny's edges (at a coarser resolution) are input to the MRF integration scheme.

*Zero-Crossings*

Because the derivative operation is ill-posed, we need to filter the resultant data through an appropriate low-pass filter [Torre and Poggio, 1985]. The filter of choice (but not the only possibility!) is a Gaussian at a suitable spatial scale. An interesting, simple implementation of Gaussian convolution relies on the binomial approximation to the Gaussian distribution. This algorithm requires only integer addition, shifting, and local communication on the 2-D mesh, so it can be implemented on a simple 2-D mesh architecture (such as the NEWS network on the Connection Machine).

The Laplacian of a Gaussian is often approximated by the difference of Gaussians. The Laplacian of a Gaussian can also be computed by convolution with a Gaussian followed by convolution with a discrete Laplacian; we have implemented both on the Connection Machine. To detect zero-crossings, the computation at each pixel need only examine the sign bits of neighboring pixels.

*Canny Edge Detection*

The Canny edge detector is often used in image understanding. It is based on directional derivatives, so it has improved localization. The Canny edge detector on the Connection Machine consists of the following steps:

- Gaussian smoothing
- Directional derivative
- Non-maximum suppression
- Thresholding with hysteresis.

Gaussian filtering, as described above, is a local operation. Computing directional derivatives is also local, using a finite difference approximation referencing only local neighbors in the image grid.

*Non-maximum Suppression*

Non-maximum suppression selects as edge candidates those pixels for which the gradient magnitude is maximal in the direction of the gradient. This involves interpolating the gradient magnitude between each of two pairs of adjacent pixels among the eight neighbors of a pixel, one forward in the gradient direction, one backward. However, it may not be critical to use interpolation, in which case magnitudes of neighboring values can be directly compared.

*Thresholding with Hysteresis*

Thresholding with hysteresis eliminates weak edges due to noise, using the threshold, while connecting extended curves over small gaps using hysteresis. Two thresholds are computed, *low* and *high*, based on an estimate of the noise in the image brightness. The non-maximum suppression step selects those pixels where the gradient magnitude is maximal in the direction of the gradient. In the thresholding step, all selected pixels with gradient magnitude below *low* are eliminated. All pixels with values above *high* are considered as edges. All pixels with values between *low* and *high* are edges if they can be connected to a pixel above *high* through a chain of pixels above *low*. All others are eliminated.

This is a spreading activation operation; it propagates information along a set of connected edge pixels. The algorithm iterates, in each step marking as *edge* pixels any *low* pixels adjacent to *edge* pixels. When no pixels change state, the iteration terminates, taking $O(m)$ steps, a number proportional to the length $m$ of the longest chain of *low* pixels which eventually become *edge* pixels. The running time of this operation can be reduced to $O(\log m)$, using *distance doubling*.

*Noise Estimation*

Estimating noise in the image can be performed by analyzing a histogram of the gradient magnitudes. Most computational implementations of this step perform a global analysis of the gradient magnitude distribution, which is essentially non-local; we have had success with a Connection Machine implementation using local histograms. The thresholds used in Canny edge detection depend on the final task for which the edges are used. A conservative strategy can use an arbitrary low threshold to eliminate the need for the costly processing required to accumulate a histogram. Where a more precise estimate of noise is needed, it may be possible to find a scheme that use a coarse estimate of the gradient magnitude distribution, with minimal global communication.

### 4.2. Stereo

The Drumheller-Poggio parallel stereo algorithm [Drumheller and Poggio, 1986] runs as part of the Vision Machine. Disparity data produced by the algorithm comprise one of the inputs to the MRF-based integration stage of the Vision Machine. We are exploring various extensions of the algorithm, as well as the possible use of feedback from the integration stage. In this section, we will review the algorithm briefly, then proceed to a discussion of current research.

The stereo algorithm runs on the Connection Machine system with good results on natural scenes in times that are typically on the order of one second. The stereo algorithm is presently being extended in the context of the Vision Machine project.

#### 4.2.1. Drumheller-Poggio Stereo Algorithm

Stereo matching is an ill-posed problem [see Bertero et.al., 1987] that cannot be solved without taking advantage of natural constraints. The *continuity constraint* [see, for instance, Marr and Poggio, 1976] asserts that the world consists primarily of piecewise smooth surfaces. If the scene contains no transparent objects, then the *uniqueness constraint* applies: there can be only one match along the left or right lines of sight. If there are no narrow occluding objects, the *ordering constraint* [Poggio and Yuille, 1984] holds: any two points must be imaged in the same relative order in the left and right eyes.

The specific *a priori* assumption on which the algorithm is based is that the disparity – that is, the depth of the surface – is locally constant in a small region surrounding a pixel. It is a restrictive assumption which, however, may be a satisfactory *local* approximation in many cases (it can be extended to more general surface assumptions in a straightforward way but at high computational cost). Let $E_L(x, y)$ and $E_R(x, y)$ represent

the left and the right image of a stereo pair or some transformation of it, such as filtered images or a map of the zero-crossings in the two images (more generally, they can be maps containing a feature vector at each location $(x, y)$ in the image).

We look for a discrete disparity $d(x, y)$ at each location $x, y$ in the image that minimizes

$$\|E_L(x, y) - E_R(x + d(x, y), y)\|_{\text{patch}_i} \qquad (1)$$

where the norm is a summation over a local neighborhood centered at each location $(x, y)$; $d(x)$ is assumed constant in the neighborhood. Equation (1) implies that we should look at each $(x, y)$ for $d(x, y)$ such that

$$\int_{\text{patch}_i} (E_L(x, y) E_R(x + d(x, y), y))^2 dx dy \qquad (2)$$

is maximized.

The algorithm that we have implemented on the Connection Machine is actually somewhat more complicated, since it involves geometric constraints that affect the way the maximum operation is performed (see Drumheller and Poggio, 1986). The implementation currently used in the Vision Machine at the AI Lab uses the maps of Canny's edges obtained from each image for $E_L$ and $E_R$.

In more detail, the algorithm is composed of the following steps:

1) Compute features for matching.

2) Compute potential matches between features.

3) Determine the degree of continuity around each potential match.

4) Choose correct matches based on the constraints of continuity, uniqueness, and ordering.

Potential matches between features are computed in the following way. Assuming that the images are registered so that the epipolar lines are horizontal, the stereo matching problem becomes one-dimensional: an edge in the left image can match any of the edges in the corresponding horizontal scan line in the right image. Sliding the right image over the left image horizontally, we compute a set of *potential match planes*, one for each horizontal disparity. Let $p(x, y, d)$ denote the value of the $(x, y)$ entry of the potential match plane at disparity $d$. We set $p(x, y, d) = 1$ if there is an edge at location $(x, y)$ in the left image and a compatible edge at location $(x - d, y)$ in the right image; otherwise, set $p(x, y, d) = 0$. In the case of the DOG edge detector, two edges are compatible if the sign of the convolution for each edge is the same.

To determine the degree of continuity around each potential match $(x, y, d)$, we compute a *local support score* $s(x, y, d) = \sum_{patch} p(x, y, d)$, where *patch* is a small neighborhood of $(x, y, d)$ within the $dth$ potential match plane. In effect, nearby points in *patch* can "vote" for the disparity $d$. The score $s(x, y, d)$ will be high if the continuity constraint is satisfied near $(x, y, d)$, i.e., if *patch* contains many votes. This step corresponds to the integral over the patch in Equation (2).

Finally, we attempt to select the correct matches by applying the uniqueness and ordering constraints (see above). To apply the uniqueness constraint, each match suppresses all other matches along the left and right lines of sight with weaker scores. To enforce the ordering constraint, if two matches are not imaged in the same relative order in left and right views, we discard the match with the smaller support score. In effect, each match suppresses matches with lower scores in its forbidden zone [Poggio and Yuille, 1984]. This step corresponds to choosing the disparity value that maximizes the integral of Equation (2).

### 4.2.2. Improvements

Using this algorithm as a base, we are exploring the following topics:

*Detection of Depth Discontinuities*

The Marr-Poggio continuity constraint is both a strength and a weakness of the stereo algorithm. Favoring continuous disparity surfaces reduces the solution space tremendously, but also tends to smooth over depth discontinuities present in the scene. Consider what happens near a linear depth discontinuity, say a point near the edge of a table viewed from above. The square local support neighborhood for the point will be divided between points on the table and points on the floor; thus, almost half of the votes will be for the wrong disparity.

One solution to this problem is feedback from the MRF integration stage. We can take the depth discontinuities located by the integration stage (using the results from a first pass of the stereo algorithm, among other inputs) and use them to restrict the local support neighborhoods so that they do not span discontinuities. In the example mentioned above, the support neighborhood would be trimmed to avoid crossing the discontinuity between the table and the floor, and thus would not pick up spurious votes from the floor.

We can also try to locate discontinuities by examining intermediate results of the stereo algorithm.

Consider a histogram of votes vs. disparity for the table/floor example. For a support region centered near the edge of the table, we expect to see two strong peaks: one at the disparity of the floor, and the other at the disparity of the table. Therefore a bimodal histogram is strong evidence for the presence of a discontinuity.

These two ideas can be used in conjunction. Discontinuity detection within stereo can take advantage of the extra information provided by the vote histograms. By passing better depth data (and perhaps candidate discontinuity locations) to the integration stage, we improve the detection of discontinuities at the higher level.

*Improving the Stereo Matcher*

The original Drumheller-Poggio algorithm matched DOG zero-crossings, where the local support score counted the number of zero-crossings in the left image patch matching edges in the right image patch, at a given disparity. We have modified the matcher in a variety of ways.

1) Canny edges. The matcher now uses edges derived by a parallel implementation of the Canny edge detector [Canny, 1983; Little et al., 1987] rather than DOG zero-crossings, for better localization.

2) Gradient direction constraint. We allow two Canny edges to match only if the associated brightness gradient directions are aligned within a parameterized tolerance. This is analogous to the restriction in the Marr-Poggio-Grimson stereo algorithm [Grimson,1981], where two zero-crossings can match only if the directions of the DOG gradients are approximately equal. Matching gradient orientations is a tighter constraint than matching the sign of the DOG convolution. Furthermore, the DOG sign is numerically unstable for horizontally oriented edges.

3) The scores are now normalized to take into account the number of edges in the left and right image patches eligible to match, so that patches with high edge densities do not generate artificially high scores. We plan to change the matcher so that edges that fail to match would count as negative evidence by reducing the support score, but this has not yet been implemented.

In the near future, we will explore matching brightness values as well as edges, using a cross-correlation approach similar to that of Little, Bülthoff and Poggio [1987] for motion estimation [Gillett, in preparation].

*Identifying Areas that are Outside of the Matcher's Disparity Range*

The stereo algorithm searches a limited disparity range, selected manually. Every potential match in the scene (an edge with a matching edge at some disparity) is assigned the in-range disparity with the highest score, even though the correct disparity may be out of range. How can we tell when an area of the scene is out of range?

The most effective approach that we have attempted to date is to look for regions with low matching scores. Two patches that are incorrectly matched will, in general, produce a low matching score.

### 4.2.3. Memory-Based Registration and Calibration

Registration of the image pair for the stereo algorithm is done by presenting to the system a pattern of dots, roughly on a sparse grid, at the distance around which stereo has to operate. The registration is accomplished using a warping computed by matching the dots from the left and right images. The dots are sparse enough that matching is unambiguous. The matching defines a warping vector for each dot; at other points the warping is computed by bilinear interpolation of the two components of warping vectors. The warping necessary for mapping the right image onto the left image is then stored. Prior to stereo-matching, the right image is warped according to the pre-stored addresses by sending each pixel in the right image to the processor specified in the table.

The warping table corrects for deformations, including those due to vertical disparities and rotations, those due to the image geometry (errors in the alignment of the cameras, perspective projection, errors introduced by the optics, etc.) We plan to store several warping tables for each of a few convergence angles of the two cameras (assuming symmetric convergence). We conjecture that simple interpolation can yield sufficiently accurate warping tables for fixation angles intermediate to the ones stored. Notice that these tables are independent of the position of the head. Absolute depth is not the concern here (we are not using it in our present Vision Machine), but it could easily be recovered from knowledge of the convergence angle. Notice also that the whole registration scheme has the flavor of a learning process. Convergence angles are inputs and warping tables are the outputs of the modules; the set of angles, together with the associated warping tables, represent the set of input-output examples. The system can "generalize" by interpolating between warp-

ing tables and providing the warping corresponding to a vergence angle that does not appear in the set of "examples". Calibration of disparity to depth could be done in a similar way.

### 4.3 Motion

The motion algorithm computes the optical flow field, a vector field which approximates the projected motion field. The procedure produces sparse or dense output, depending on whether it uses edge features or intensities. The algorithm assumes that image displacements are small, within a range $(\pm\delta, \pm\delta)$. It is also assumed that the optical flow is locally constant in a small region surrounding a point. This assumption is strictly only true for translational motion of 3–D planar surface patches parallel to the image plane. It is a restrictive assumption which, however, may be a satisfactory *local* approximation in many cases. Let $E_t(x,y)$ and $E_{t+\Delta t}(x,y)$ represent transformations of two discrete images separated by time interval $\Delta t$, such as filtered images or a map of the brightness changes in the two images (more generally, they can be maps containing a feature vector at each location $(x,y)$ in the image) [Kass, 1986; Nishihara, 1984].

We look for a discrete motion displacement $\underline{v} = (v_x, v_y)$ at each location $x, y$ in the image that minimizes

$$\|E_t(x,y) - E_{t+\Delta t}(x+v_x\Delta t, y+v_y\Delta t)\|_{\text{patch}_i} = \min \quad (3)$$

where the norm is a summation over a local neighborhood centered at each location $(x,y)$; $\underline{v}(x,y)$ is assumed constant in the neighborhood. Equation (3) implies that we should look at each $(x,y)$ for $\underline{v} = (v_x, v_y)$ such that

$$\int_{\text{patch}_i} (E_t(x,y) - E_{t+\Delta t}(x+v_x\Delta t, y+v_y\Delta t))^2 dx dy \quad (4)$$

is minimized. Alternatively, one can maximize the negative of the integrated result. Equation (4) represents the sum of the pointwise squared differences between a patch in the first image centered around the location $(x,y)$ and a patch in the second image centered around the location $(x+v_x\Delta t, y+v_y\Delta t)$.

This algorithm can be translated easily into the following description. Consider a network of processors representing the result of the integrand in Equation (4). Assume for simplicity that this result is either 0 or 1 (this is the case if $E_t$ and $E_{t+\Delta t}$ are binary feature maps). The processors hold the result of differencing (taking the logical "exclusive or") the right and left image map for different values of $(x,y)$ and $v_x, v_y$. The next stage, corresponding exactly to the integral operation over the patch, is for each processor to summate the total (2) in an $(x,y)$ neighborhood at the same disparity. Note that this summation operation is efficiently implemented on the Connection Machine using *scan* computations. Each processor thus collects a vote indicating support that a patch of surface exists at that displacement. The algorithm iterates over all displacements in the range $(\pm\delta, \pm\delta)$, recording the values of the integral (2) for each displacement. The last stage is to choose $\underline{v}(x,y)$ among the displacements in the allowed range that maximizes the integral. This is done by an operation of "non-maximum suppression" across velocities out of the finite allowed set: at the given $(x,y)$, the processor is found that has the maximum vote. The corresponding $\underline{v}(x,y)$ is the velocity of the surface patch found by the algorithm. The actual implementation of this scheme can be simplified so that the "non-maximum suppression" occurs during iteration over displacements, so that no actual table of summed differences over displacements need be constructed. In practice, the algorithm has been shown to be effective both for synthetic and natural images using different types of features or measurements on the brightness data, including edges (both zero-crossings of the Laplacian of Gaussian and Canny's method), which generate sparse results along brightness edges, or brightness data directly, or the Laplacian of Gaussian or its sign, which generate dense results. Because the optical flow is computed from quantities integrated over the individual patches, the results are robust against the effects of uncorrelated noise.

The comparison stage employs patchwise cross-correlation, which exploits local constancy of the optical flow (the velocity field is guaranteed to be constant for translations parallel to the image plane of a planar surface patch; it is a cubic polynomial for arbitrary motion of a planar surface [see Waxman, 1986; Little et.al., 1987]. Experimentally, we have used zero-crossings, the Laplacian of Gaussian filtered image, its sign, and the smoothed brightness values, with similar results. It is interesting that methods *superficially* so different (edge-based and intensity-based) give such similar results. As we mentioned earlier, this is not surprising. There are theoretical arguments that support, for instance, the equivalence of cross-correlating the sign bit of the Laplacian filtered image and the Laplacian filtered image itself. The argument is based on the following theorem [see Little, Bülthoff and Poggio, in preparation], which is a slight reformulation of a well-known theorem.

*Theorem*

If $f(x,y)$ and $g(x,y)$ are zero mean jointly normal processes, their cross-correlation is determined fully by the correlation of the sign of $f$ and of the sign of $g$ ( and determines it). In particular

$$R_{\tilde{f},\tilde{g}} = \frac{2}{\pi} arcsin(R_{f,g})$$

where $\tilde{f} = sign\ f$ and $\tilde{g} = sign\ g$

Thus, cross-correlation of the sign bit is exactly equivalent to cross-correlation of the signal itself (for Gaussian processes). Notice that from the point of view of information, the sign bit of the signal is completely equivalent to the zero-crossing of the signal. Nishihara first used patchwise cross-correlation of the sign bit of DOG filtered images [Nishihara, 1984], and has implemented it more recently on real-time hardware [Nishihara et.al., 1988].

The existence of discontinuities can be detected in optical flow, as in stereo, both during computation and by processing the resulting flow field. The latter field is input to the MRF integration stage. During computation, discontinuities in optical flow arising from occlusions are indicated by low normalized scores for the chosen displacement.

### 4.4. Color

The color algorithm that we have implemented is a very preliminary version of a module that should find the boundaries in the surface spectral reflectance function, that is, discontinuities in the surface color. The algorithm relies on the idea of *effective illumination* and on the *single source* assumption, both introduced by Hurlbert and Poggio [see Poggio et.al., 1985].

The single source assumption states that the illumination may be separated into two components, one dependent only on wavelength and one dependent only on spatial coordinates, and generally holds for illumination from a single light source. It allows us to write the image irradiance equation for a Lambertian world as

$$I^{\nu} = k^{\nu} E(x,y) \rho^{\nu}(x,y) \qquad (5)$$

where $I^{\nu}$ is the image irradiance in the $\nu th$ spectral channel ($\nu = red, green, blue$), $\rho^{\nu}(x,y)$ is the surface spectral reflectance (or albedo) and the effective illumination $E(x,y)$ absorbs the spatial variations of the illumination and the shading due to the 3D shape of surfaces ($k^{\nu}$ is a constant for each channel and depends only on the luminant). A simple segmentation algorithm is then obtained by considering the equation

$$H(x,y) = \frac{I^r}{I^r + I^g} = \frac{k^r\ \rho^r}{k^r \rho^r\ +\ k^g \rho^g} \qquad (6)$$

which changes only when $\rho^r$ or $\rho^g$ or both change. Thus $H$, which is piecewise constant, has discontinuities that mark changes in the surface albedo, independently of changes in the effective illumination.

The quantity $H(x,y)$ is defined almost everywhere, but is typically noisy. To counter the effect of noise, we exploit the prior information that $H$ should be piecewise constant with discontinuities that are themselves continuous, non-intersecting lines. As we will discuss later, this restoration step is achieved by using a MRF model. This algorithm works only under the restrictive assumption that specular reflections can be neglected. Hurlbert [1988] discusses in more detail the scheme outlined here and how it can be extended to more general conditions.

### 4.5. Texture

The texture algorithm is a greatly simplified parallel version of the texture algorithm developed by Voorhees and Poggio [1987]. Texture is a scalar measure computed by summation of texton densities over small regions surrounding every point. Discontinuities in this measure can correspond to occlusion boundaries, or orientation discontinuities, which cause foreshortening. Textons are computed in the image by simple approximation to the methods presented in Voorhees and Poggio [1987]. For this example, the textons are restricted to blob-like regions, without regard to orientation selection.

To compute textons, the image is first filtered by a Laplacian of Gaussian filter at several different scales. The smallest scale selects the textural elements. The Laplacian of Gaussian image is then thresholded at a non-zero value to find the regions which comprise the blobs identified by the textons. The result is a binary image with non-zero values only in the areas of the blobs. A simple summation counts the density of blobs, the portion of the summation region covered by blobs, in a small area surrounding each point. This operation effectively measures the density of blobs at the small scale, while also counting the presence of blobs caused by large occlusion edges at the boundaries of textured regions. Contrast boundaries appear as blobs in the Laplacian of Gaussian image. To remove their effect, we use the Laplacian of Gaussian image at a slightly coarser scale. Blobs caused by the texture at the fine scale do not appear at this coarser scale, while the contrast boundaries, as well as all other blobs at coarser scales, remain. This coarse blob image filters the fine

blobs – blobs at the coarser scale are removed from the fine scale image. Then, summation, whether with a simple scan operation, or Gaussian filtering, can determine the blob density at the fine scale only. This is one example where multiple spatial scales are used in the present implementation of the Vision Machine.

## 5. The Integration Stage and MRF

Whereas it is reasonable that combining the evidence provided by multiple cues – for example, edge detection, stereo and color – should provide a more reliable map of the surfaces than any single cue alone, it is not obvious how this integration can be accomplished. The various physical processes that contribute to image formation – *surface depth, surface orientation, albedo (Lambertian and specular component), illumination* – are coupled to the image data, and therefore to each other, through the imaging equation. The coupling is, however, difficult to exploit in a robust way, since it depends critically on the reflectance and imaging models. We argue that the coupling of the image data to the surface and illumination properties is of a more qualitative and robust sort at locations in which image brightness changes sharply and surface properties are discontinuous, in short, at edges. The intuitive reason for this is that at discontinuities, the coupling between different physical processes and the image data is robust and qualitative. For instance, a depth discontinuity usually originates a brightness edge in the image, and a motion boundary often corresponds to a depth discontinuity (and an brightness edge) in the image. This view suggests the following integration scheme for restoring the data provided by early modules. The results provided by stereo, motion and other visual cues are typically noisy and sparse. We can improve them by exploiting the fact that they should be smooth, or even piecewise constant (as in the case of the albedo), between discontinuities. We can exploit *a priori* information about generic properties of the discontinuities themselves: for instance, that they usually are continuous and non intersecting.

The idea is then to detect discontinuities in each cue, say depth, simultaneously with the approximation of the depth data. The detection of discontinuities is helped by information on the presence and type of discontinuities in the surfaces and surface properties (see Figure 1), which are coupled to the brightness edges in the image.

Notice that reliable detection of discontinuities is critical for a vision system, since discontinuities are often the most important locations in a scene: depth discontinuities, for example, normally correspond to the boundaries of an object or an object part. The idea is thus to couple different cues through their discontinuities and to use information from several cues simultaneously to help refine the initial estimation of discontinuities, which are typically noisy and sparse.

How can this be done? We have chosen to use the machinery of Markov Random Fields (MRFs), initially suggested for image processing by Geman and Geman [1984]. In the following we will give a brief, informal outline of the technique and of our integration scheme. More detailed information about MRFs can be found in Geman and Geman [1984] and Marroquin et.al. [1987]. Gamble and Poggio [1987] describe an earlier version of our integration scheme and its implementation, outlined in the next section.

### 5.1. MRF Models

Consider the prototypical problem of approximating a surface given sparse and noisy data (depth data), on a regular 2D lattice of sites. We first define the prior probability of the class of surfaces we are interested in. The probability of a certain depth at any given site in the lattice depends only upon neighboring sites (the Markov property). Because of the Clifford-Hammersley theorem, the prior probability is guaranteed to have the Gibbs form

$$P(f) = \frac{1}{Z} e^{-\frac{U(f)}{T}} \tag{7}$$

where $Z$ is a normalization constant, $T$ is called temperature, and $U(f) = \sum_C U_C(f)$ is an energy function that can be computed as the sum of local contributions from each neighborhood. The sum of the *potentials*, $U_C(X)$, is over the neighborhood's *cliques*. A clique is either a single lattice site or a set of lattice sites such that any two sites belonging to it are neighbors of one another. Thus $U(f)$ can be considered as the sum over the possible configurations of each neighborhood [see Marroquin et.al., 1987]. As a simple example, when the surfaces are expected to be smooth, the prior probability can be given in terms of

$$U_c(f) = (f_i - f_j)^2 \tag{8}$$

where $i$ and $j$ are neighboring sites (belonging to the same clique).

If a model of the observation process is available (i.e., a model of the noise), then one can write the conditional probability $P(g/f)$ of the sparse observation $g$ for any given surface $f$. Bayes Theorem then allows one to write the posterior distribution

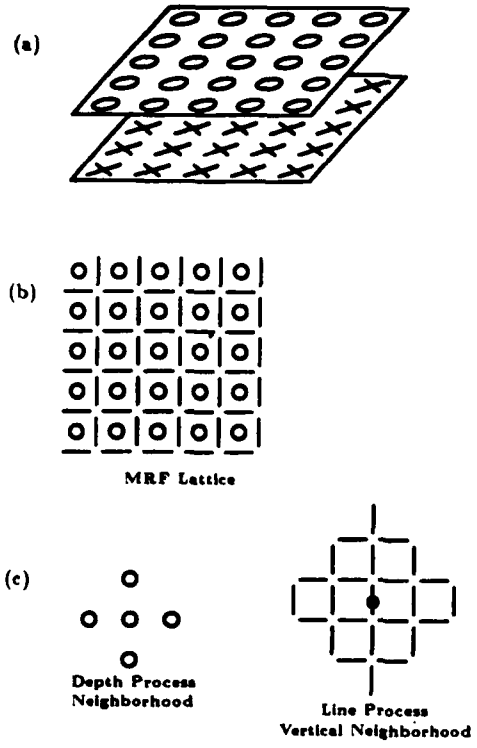



Figure 5: The MRF lattice

$$P(f/g) = \frac{1}{Z} e^{\frac{-U(f/g)}{T}} \qquad (9)$$

In the simple earlier example, we have (for Gaussian noise)

$$U(f/g) = \sum_C \alpha \gamma_i (f_i - g_i)^2 + (f_i - f_j)^2 \qquad (10)$$

where $\gamma_i = 1$ only where data are available. More complicated cases can be handled in a similar manner.

The posterior distribution cannot be solved analytically, but sample distributions with the probability distribution of Equation (3) can be obtained using Monte Carlo techniques such as the Metropolis algorithm. These algorithms sample the space of possible surfaces according to the probability distribution $P(f/g)$ that is determined by the prior knowledge of the allowed class of surfaces, the model of noise, and the observed data. In our implementation, a highly parallel computer generates a sequence of surfaces from which, for instance, the surface corresponding to the maximum of $P(f/g)$ can be found. This corresponds to finding the global minimum of $U(f/g)$ (simulated annealing is one of the possible techniques). Other criteria can be used: Marroquin [1985] has shown that the average surface $f$ under the posterior distribution is often a better estimate which can be obtained more efficiently simply by finding the average value of $f$ at each lattice site.

One of the main attractions of MRFs is that the prior probability distribution can be made to embed more sophisticated assumptions about the world. Geman and Geman [1984] introduced the idea of another process, the line process, located on the dual lattice (see Figure 5), and representing explicitly the presence or absence of discontinuities that break the smoothness assumption (Equation (2)). The associated prior energy then becomes

$$U_C(f) = (f_i - f_j)^2 (1 - l_i^j) + \beta V_C(l_i^j) \qquad (11)$$

where $l$ is a binary line element between site $i, j$. $V_C$ is a term that reflects the fact that certain configurations of the line process are more likely than others to occur. In our world, depth discontinuities are usually themselves continuous, non-intersecting, and rarely isolated joints. These properties of physical discontinuities can be enforced locally by defining an appropriate set of energy values $V_C(l)$ for different configurations of the line process in the neighborhood of the site (notice that the assignment of zero energy values to the non-central cliques mentioned in Gamble and Poggio [1987] is wrong, as pointed out to us by Tail Symchony).

### 5.2. Organization of Integration

It is possible to extend the energy function of Equation (5) to accommodate the interaction of more processes and of their discontinuities. In particular, we have extended the energy function to couple several of the early vision modules (depth, motion, texture and color) to brightness edges in the image. This is a central point in our integration scheme: brightness edges guide the computation of discontinuities in the physical properties of the surface, thereby coupling surface depth, surface orientation, motion, texture and color, each to the image brightness data and to each other. The reason for the role of brightness edges is that changes in surface properties usually produce large brightness gradients in the image. It is exactly for this reason that edge detection is so important in both artificial and biological vision.

The coupling to brightness edges may be done by replacing the term $V_C(l_i^j)$ in the last equation with the term

$$V(l,e) = g(e_i^j, V_C(l_i^j)) \qquad (12)$$

with $e_i^j$ representing a measure of the presence of an brightness edge between site $i,j$. The term $g$ has the effect of modifying the probability of the line process configuration depending on the brightness edge data ($V(l,e) = -log\ p(l/e)$). This term facilitates formation of discontinuities (that is, $l_i^j$) at the locations of brightness edges. Ideally, the brightness edges (and the neighboring image properties) activate, with different probabilities, the different surface discontinuities (see Figure 1) which in turn are coupled to the output of stereo, motion, color, texture, and possibly other early algorithms.

We have been using the MRF machinery with prior energies like that given in Equations (11) and (12) (see also Figure 1) to integrate edge brightness data with stereo, motion and texture information on the MIT Vision Machine System.

We should emphasize that our present implementation represents a subset of the possible interactions shown in Figure 1, itself only a simplified version of the organization of the likely integration process. The system will be improved in an incremental fashion, including pathways not shown in Figure 1, such as feed-backs from the results of integration into the matching stage of the stereo and motion algorithms.

### 5.3. Algorithms: Deterministic and Stochastic

A few disclaimers are in order here. We have chosen to use MRF models because of their generality and theoretical attractiveness. This does not imply that stochastic algorithms must be used. For instance, in the cases in which the MRF model reduces to standard regularization [Marroquin et.al., 1987] and the data are given on a regular grid, the MRF formulation leads not only to a purely deterministic algorithm, but also to a convolution filter.

We expect that during our research we will define deterministic algorithms that are either equivalent to a MRF formulation, or are a good approximation to the stochastic Monte Carlo algorithms. More specifically, we expect that the probabilistic formulation of MRF is in a sense too general, and therefore too inefficient. Remember that MRF models are quite general (for instance, regularization can be regarded from a probabilistic point of view as an instance of MRF).

## 6. Illustrative Results

Figures 7 and 8 show the results of the Vision Machine applied to the scene in Figure 6 and some of the intermediate steps. Figure 7 show the brightness edges computed by the Canny algorithm at two different spatial scales ($\sigma = 2.5$ and $\sigma = 4$). We show neither the stereo pair nor the motion sequence in which the teddy bear was rolling slightly on his back from one frame to the next. The results given by the stereo, motion, texture and color algorithms, after an initial smoothing to make them dense [see Gamble and Poggio, 1987], are shown in the first column on the left of Figure 8 (from top to bottom). They represent the input to the MRF machinery that integrates each of those data sets with the brightness edges. The color algorithm uses the edges at the coarser resolution, since we want to avoid detecting texture marks on the surface; the other cues are integrated with the Canny edges at a smaller scale ($\sigma = 2.5$). The central column of Figure 8 shows the reconstructed depth, color (the quantity $H$ defined earlier), texture and motion flow; the left column show the discontinuities found by the MRF machinery in each of the cues. Processing of the stereo output finds depth discontinuities in the scene (mainly the outlines of the teddy, plus a fold of a wet suit protruding outward). Motion discontinuities are found by the MRF machinery with help from brightness edges. The color boundaries show regions of constant surface color, independently of its shading: notice, for instance, that brightness edges inside the teddy bear, due to shading, do not appear as color edges (the color images were taken from a different camera). The texture boundaries correspond quite well to different textured surfaces.

Figure 9 shows that the union of the discontinuities in depth and motion for the scene of Figure 6 gives a rather good "cartoon" of the original scene. At the same time, our integration algorithm achieves a preliminary classification of the brightness edges in the image, in terms of their physical origin. A more complete classification will be achieved by the full scheme of Figure 1: the lattices at the top classify the different types of discontinuities in the scene. The set of such discontinuities in the various physical processes should represent a good set of data for later recognition stages.

## 7. The Future

The Vision Machine is evolving rapidly. We plan to add other early vision algorithms (such as shape-from-shading) and to develop further the ones already

Figure 6: Grey-level image of a natural scene processed by the Vision Machine

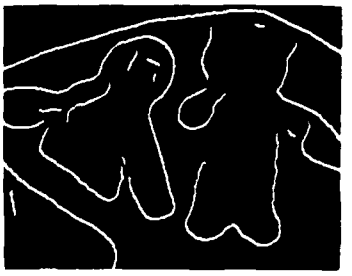


Figure 7: Canny edges of the image in Figure 6

implemented (especially color and texture). Most of the future effort will be directed towards a more satisfactory integration: we will define and implement a scheme of the type shown in Figure 1. Finding the correct values of the parameter is critical for the practical success of the MRF technique; thus we will attempt to find useful solutions to the parameter estimation problem, an issue strictly related to learning from examples. An important step in the very near future will be the implementation of recognition algorithms operating on the output of the integration stage.

### 7.1. Towards Recognition

The output of the integration stage provides a set of edges labeled in terms of physical discontinuities of the surface properties. They should represent a good input to a model-based recognition algorithm like the ones described by Dan Huttenlocher and by Todd Cass in these Proceedings. In particular, we are interfacing the Vision Machine as implemented so far with the Cass algorithm. Initially, we will use only discontinuities for recognition; later we will use also the information provided by the MRFs on the surface properties *between* discontinuities. Notice that in the full system we may have several visual routines [see Poggio et.al., these Proceedings) operating also on the maps of physical discontinuities and performing task-dependent grouping operations before recognition.

### 7.2. Learning and Parameter Estimation

Using the MRF model involves an energy function which has several free parameters, in addition to the many possible neighborhood systems. The values of these parameters determine a distribution over the configuration-space to which the system converges, and the speed of convergence. Thus rigorous methods for estimating these parameters are essential for the practical success of the method and for meaningful results. In some cases, parameters can be learned from the data: e.g., texture parameters [Geman and Graffigne, 1987], or neighborhood parameters (for which a cellular automaton model may be the most convenient for the purpose of learning). There are general statistical methods which can be used for parameter estimation:

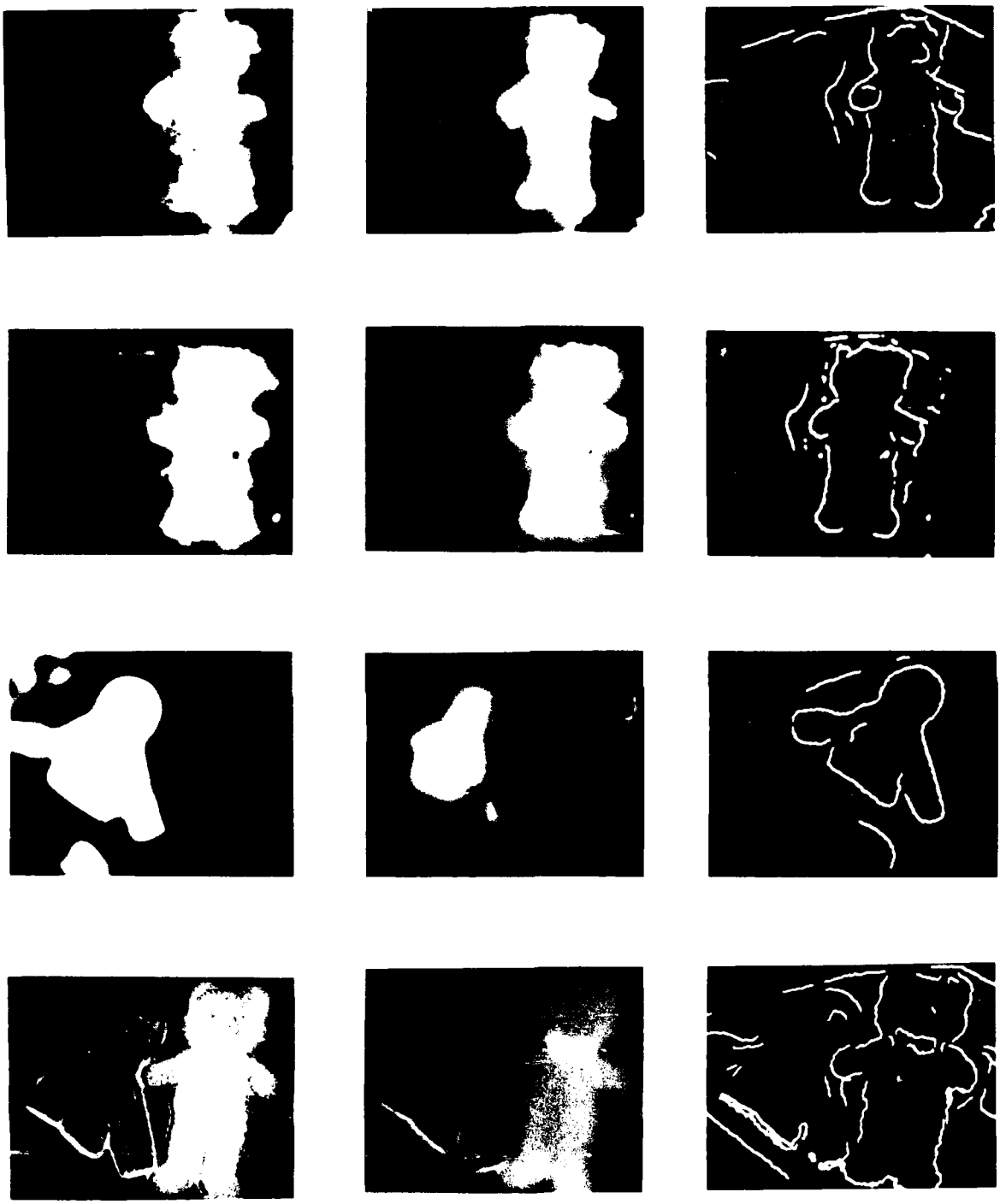


Figure 8: MRF results for stereo, motion, texture and color

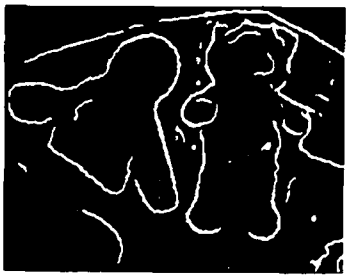


Figure 9: Union of depth and motion discontinuities

- A maximum likelihood estimate – one can use the indirect iterative EM algorithm [Dempster et.al., 1977], which is most useful for maximum likelihood estimation from incomplete data [see Marroquin, 1987 for a special case]. This algorithm involves the iterative maximization (over the parameter space) of the expected value of the likelihood function given that the parameters take the values of their estimation in the previous iteration. Alternatively, a search constrained by some statistics for a minimum of an appropriate merit function may be employed [see Marroquin, 1987].

- A smoothing (regularization) parameter can be estimated using the methods of cross-validation or unbiased risk, to minimize the mean square error. In cross-validation, an estimate is obtained omitting one data point. The goal is to minimize the distance between the predicted data point (from the estimate above with the point omitted) and the actual value, for all points.

In the case of Markov Random Fields, some more specific approaches are appropriate for parameter estimation:

1) Besag [1974] suggested conditional maximum likelihood estimation using coding methods, maximum likelihood estimation with unilateral approximations on the rectangular lattice, or "maximum pseudolikelihood" – a method to estimate parameters for homogeneous random fields [see Geman and Graffigne, 1987].

2) For the MPM estimator, where a fixed temperature is yet another parameter to be estimated, one can try to use the physics behind the model to find a temperature with as little disorder as possible and still reasonable time of convergence to equilibrium (e.g., away from "phase-transition").

An alternative asymptotic approach can be used with smoothing (regularization) terms: instead of estimating the smoothing parameter, let it tend to 0 as the temperature tends to 0, to reduce the smoothing close to the final configuration [see Geman and Geman, 1987].

In summary, we plan to explore three distinct stages for parameter estimation in the integration stage of the Vision Machine:

- *Modeling* (from the physics of surfaces, of the imaging process and of the class of scenes to be analyzed and the tasks to be performed) and the form of the prior and of some conditional probabilities involved (e.g., the type of physical edges from properties of the measurements, such as characteristics of the brightness data). Range of allowed parameter values may also be established at this stage (e.g., minimum and maximum brightness value in a scene, depth differences, positivity of certain measurements, distribution of expected velocities, reflectance properties, characteristics of the illuminant, etc.).

- *Estimating* of parameter values from set of examples in which data and desired solution are given. This is a *learning stage*. We may have to use days of CM time and, at least initially, synthetic images to do this.

- *Tuning* of some of the parameters directly from the data (by using EM algorithm, cross-validation, Besag's work, or various types of heuristics).

The dream is that at some point in the future the Vision Machine will run all the time, day and night, looking about and learning on its own to see better and better.

### 7.3. Fast Vision: The Role of Time Smoothness

The present version of the Vision Machine processes only isolated frames. Even our motion algorithm takes as input simply a sequence of two images. The reason for this is, of course, limitations in raw speed. We cannot perform all of the processing we do at video rate (say, 30 frames per second), though this goal is certainly within present technological capabilities. If we could process frames at video rate, we could exploit constraints in the time dimension similar to the ones we are already exploiting in the space domain. Surfaces and even the brightness array itself do not usually change too much from frame to frame. This is a constraint of

smoothness in time, which is valid almost everywhere, but not *across* discontinuities in time. Thus one may use the same MRF technique, applied to the output of stereo, motion, color, and texture, and enforcing continuity in time (if there are no discontinuities), that is, exploiting the redundancy in the sequence of frames.

We believe that the surface reconstructed from a stereo pair usually does not need to be recomputed completely when the next stereo pair is taken a fraction of a second later. Of course, the role of the MRFs may be accomplished in this case by some more specific and more efficient deterministic method such as, for instance, a form of Kalman filtering. Notice that space-time MRFs applied to the brightness arrays would yield spatiotemporal interpolation and approximation of a kind already considered [Fahle and Poggio, 1980; Poggio, Nielsen and Nishihara, 1982; Bliss, 1985].

### 7.4. A VLSI Vision Machine?

Our Vision Machine is mostly specialized software running on a general purpose computer, the Connection Machine. This is a good system for the present stage of experimentation and development. Later, once we have perfected and tested the algorithms and the overall system, it will make sense to compile the software in silicon in order to produce a faster, cheaper, and smaller Vision Machine. We are presently planning to use VLSI technologies to develop some initial chips as a first step toward this goal. In this section, we will outline some thoughts about VLSI implementation of the Vision Machine.

*Algorithms and Hardware*

We realize that our specialized software vision algorithms are not, in general, optimized for hardware implementation. So, rather than directly "hardwiring algorithms" into standard computing circuitry, we will be investigating "algorithmic hardware" designs that utilize the local, symmetric nature of early vision problems. This will be an iterative process, as the algorithm influences the hardware design and as hardware constraints modify the algorithm.

*Degree of Parallelism*

Typical vision tasks require tremendous amounts of computing power and are usually parallel in nature. As an example, biology uses highly parallel networks of relatively slow components to achieve sophisticated vision systems. However, when implementing our algorithms into silicon integrated circuits, it is not clear what level of parallelism is necessary. While biology is able to use three dimensions to construct highly interconnected parallel networks, VLSI is limited to $2\frac{1}{2}$ dimensions, making highly parallel networks much more difficult and costly to implement. However, the electrical components of silicon integrated circuits are approximately four orders of magnitude faster than the electrochemical components of biology. This suggests that pipelined processing or other methods of time-sharing computing power may be able to compensate for the lower degree of connectivity of silicon VLSI. Clearly, the architecture of a VLSI vision system may not resemble any biological vision systems.

*Signal Representation*

Within the integrated circuit, the image data may be represented as a digital word or an analog value. While the advantages of digital computation are its accuracy and speed, digital circuits do not have as high functionality per device as analog circuits. Therefore, analog circuits should allow much denser computing networks. This is particularly important for the integration of computational circuitry and photosensors, which will help to alleviate the I/O bottleneck typically experienced whenever image data are serially transferred between Vision Machine components. However, analog circuits are limited in accuracy and are difficult to characterize and design.

The primary motivation for a VLSI implementation of our Vision Machine is to increase the computational speed and reduce the physical size of the components with the eventual goal of real-time, mobile vision systems. While the main computational engine of our Vision Machine is the Connection Machine, which is a very powerful and flexible SIMD computer, specific VLSI implementations will attempt to tradeoff computational flexibility for faster performance and higher degree of integration. A VLSI implementation of our Vision Machine can offer significant improvements in performance that would be difficult or impossible to attain by other methods. Presently, we are specifically investigating the integration of charge coupled devices for photosensing and simple parallel computations, such as binomial convolution and patchwise correlation.

### Legends

**Figure 1:** A diagram of the overall organization of the integration stage [see Gamble and Poggio, 1987 for a complementary diagram]. The output of each of the early visual cues (or algorithms) - stereo, motion, texture and color - are coupled to their own line process (the crosses), i.e., their discontinuities. They are also coupled to the discontinuities in the surface properties - occluding edges (both extremal

edges and blades), orientation discontinuities, specular edges, texture marks (including albedo discontinuities), shadow edges. The image data – mainly its intensity edges – are input to the lattices that represent the discontinuities in the physical properties of the scene. Our present implementation does not distinguish the different types of physical discontinuities: intensity edges are directly coupled to the line processes of each of the cues. The intensity edges can be completed and extended by the equivalent of a higher order MRF that reflects constraints of colinearity and continuation, and even hypotheses from the recognition stage, which uses the set of discontinuities at the top as its main input.

**Figure 2:** The Eye-Head System. See text.

**Figure 3:** See text.

**Figure 4:** See text.

**Figure 5:** A MRF lattice, consisting of depth (or color or texture) process elements (circles) with vertical and horizontal line elements (lines). The neighborhood of the depth process and of the line process are also shown, together with three of the several configurations of the line process. The cost of an isolated line process is much higher than that of a continuous line.

**Figure 6:** Grey-level image of a natural scene processed by the Vision Machine.

**Figure 7:** Canny edges of the image in Figure 6. See text.

**Figure 8:** See text.

**Figure 9:** See text.

### Reading List


Barrow, H.G. and J.M. Tenenbaum. "Recovering Intrinsic Scene Characteristics from Images," In: **Computer Vision Systems**, A. Hanson and E. Riseman (eds.), Academic Press, New York, 1978.

Bertero, M., T. Poggio and V. Torre. "Ill-Posed Problems in Early Vision," *Artificial Intelligence Laboratory Memo 924*, Massachusetts Institute of Technology, Cambridge, MA, 1986.

Besag, J. "Spatial Interaction and the Statistical Analysis of Lattice systems," *J. Roy. Stat. Soc.*, **B34**, 75-83, 1972.

Blake, A. "On the Geometric Information Obtainable from Simultaneous Observation of Stereo Contour and Shading," *Technical Report CSR-205-86*, Dept. of Computer Science, University of Edinburgh, 1986.

Blelloch, G.E. "Scans as Primitive Parallel Operations," *Proc. Intl. Conf. on Parallel Processing*, 355-362, 1987.

Bliss, J. "Velocity Tuned Spatio-Temporal Interpolation and Approximation in Vision," Master's Thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1985.

Brooks, R. "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, **RA-2**, 14-23, 1987.

Bülthoff H. and H. Mallot. "Interaction of Different Modules in Depth Perception," *Proc. First Intl. Conf. on Computer Vision*, Computer Society of the IEEE, Washington, DC, 295-305, 1987.

Bülthoff, H. and H. Mallot. "Interaction of Different Modules in Depth Perception: Stereo and Shading," *Artificial Intelligence Laboratory Memo 965*, Massachusetts Institute of Technology, Cambridge, MA, 1987.

Canny, J.F. "Finding Edges and Lines," *Artificial Intelligence Laboratory Technical Report 720*, Massachusetts Institute of Technology, Cambridge, MA, 1983.

Cornog, K.H. "Smooth Pursuit and Fixation for Robot Vision," Master's Thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1985.

Dempster, A.P., N.M. Laird and D.B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Roy. Stat. Soc.*, **B39**, 1-38, 1977.

Drumheller, M. and T. Poggio. "On Parallel Stereo," *Proc. Intl. Conf. on Robotics and Automation*, IEEE, 1986.

Fahle, M. and T. Poggio. "Visual Hyperacuity: Spatiotemporal Interpolation in Human Vision," *Proc. Roy. Soc. Lond. B*, **213**, 451-477, 1980.

Geman, D. and S. Geman. "Relaxation and Annealing with Constraints," *Complex Systems Technical Report 95*, Division of Applied Mathematics, Brown University, Providence, RI, 1987.

Geman, S. and D. Geman. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, **6**, 1984.

Geman, S. and C. Graffigne. "Markov Random Field Image Models and their Applications to Computer Vision," *Proc. Intl. Congress of Mathematicians*, preprint, A.M. Gleason (ed.), 1987.

Gamble, E. and T. Poggio. "Integration of Intensity Edges with Stereo and Motion," *Artificial Intelligence Laboratory Memo 970*, Massachusetts Institute of Technology, Cambridge, MA, 1987.

Grimson, W.E.L. **From Images to Surfaces**, The MIT Press, Cambridge, MA, 1981.

Grimson, W.E.L. "A Computational Theory of Visual Surface Interpolation," *Phil. Trans. Roy. Soc. Lond. B*, **298**, 395-427, 1982.

Grimson, W.E.L. "Binocular Shading and Visual Surface Reconstruction," *Computer Vision, Graphics and Image Processing*, **28**, 19-43, 1984.

Hildreth, E.C. **The Measurement of Visual Motion**, The MIT Press, Cambridge, MA, 1983.

Hillis, D. "The Connection Machine," Ph.D. Thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1985.

Horn, B.K.P. **Robot Vision**, The MIT Press, Cambridge, MA, 1986.

Hurlbert, A. "The Computation of Color Vision," Ph.D. Thesis, Dept. of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA, expected 1988.

Hurlbert, A. and T. Poggio. "Do Computers Need Attention?" *Nature*, **321**, 12, 1986.

Hurlbert, A. and T. Poggio. "Learning a Color Algorithm from Examples," *Artificial Intelligence Laboratory Memo 909/Center for Biological Information Processing Paper 25*, Massachusetts Institute of Technology, Cambridge, MA, 1987.

Huttenlocher, D. and S. Ullman. "Recognizing Rigid Objects by Aligning them with an Image," *Artificial Intelligence Laboratory Memo 937*, Massachusetts Institute of Technology, 1987.

Ikeuchi, K., and B.K.P. Horn. "Numerical Shape from Shading and Occluding Boundaries" *Artificial Intelligence*, **17**, 141-184, 1981.

Kender, J.R. "Shape from Texture: An Aggregation Transform that Maps a Class of Textures into Surface Orientation," *Proc. Sixth Intl. Joint Conf. on Artificial Intelligence*, Tokyo, 1979.

Kirkpatrick, S., C.D. Gelatt, Jr. and M.P. Vecchi. "Optimization by Simulated Annealing," *Science*, **220**, 1983.

Kruskal, C.P., L. Rudolph and M. Snir. "The Power of Parallel Prefix," *Proc. Intl. Conf. on Parallel Processing*, 180-185, 1985.

Lim, W. "Fast Algorithms for Labelling Connected Components in 2D Arrays," *Thinking Machines Corp. Technical Report NA86-1*, Cambridge, MA, 1986.

Little, J., Blelloch, G.E. and T. Cass. "Parallel Algorithms for Computer Vision on the Connection Machine," *Proceedings Intl. Conf. on Computer Vision*, 587-591, Los Angeles, 1987.

Little, J., Bülthoff, H. and Poggio, T. "Parallel Optical Flow Computation," *Proc. Image Understanding Workshop*, L. Bauman (ed.), Science Applications International Corp., McLean, VA, 915-920, 1987.

Little, J., Bülthoff, H. and Poggio, T. "Parallel Optical Flow Using Winner-Take-All Scheme," in preparation.

Mahoney, J.V. "Image Chunking: Defining Spatial Building Blocks for Scene Analysis," Master Thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1987. Published as *Artificial Intelligence Laboratory Technical Report 980*, 1987.

Marr, D. **Vision**, Freeman, San Francisco, 1982.

Marr, D. and E. Hildreth. "Theory of Edge Detection," *Proc. Roy. Soc. Lond. B*, **207**, 187-217, 1980.

Marr, D. and T. Poggio. "Cooperative Computation of Stereo Disparity," *Science*, **194**, 283-287, 1976.

Marr, D. and T. Poggio. "A Computational Theory of Human Stereo Vision," *Proc. Roy. Soc. Lond. B*, **204**, 301-328, 1979.

Marroquin, J.L. "Deterministic Bayesian Estimation of Markov Random Fields with Applications to Computational Vision," *Proc. First Intl. Conf. on*

*Computer Vision*, Computer Society of the IEEE, Washington, DC, 1987.

Marroquin, J.L. "Probabilistic Solutions of Inverse Problems," *Artificial Intelligence Laboratory Technical Report 860*, Massachusetts Institute of Technology, Cambridge, MA, 1985.

Marroquin, J.L. "Surface Reconstruction Preserving Discontinuities," *Artificial Intelligence Laboratory Memo 792*, Massachusetts Institute of Technology, Cambridge, MA, 1984.

Marroquin, J.L., Mitter S. and T. Poggio. "Probabilistic Solution of Ill-Posed Problems in Computational Vision," *Proc. Image Understanding Workshop*, L. Bauman (ed.), Scientific Applications International Corp., McLean, VA, 1986. A more complete version appears in *J. Amer. Stat. Assoc.*, **82**, 76-89, 1987.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller. "Equation of State Calculations by Fast Computing Machines," *J. Phys. Chem.*, **21**, 1953.

Nishihara, H.K. "PRISM: A Practical Real-Time Imaging Stereo Matcher," *Artificial Intelligence Laboratory Memo 780*, Massachusetts Institute of Technology, Cambridge, MA, 1984.

Nishihara, H.K. and P.A. Crossley, "Measuring Photolithographic Overlay Accuracy and Critical Dimensions by Correlating inarized Laplacian of Gaussian Convolutions," *IEEE Trans. Pattern Matching and Machine Intell.*, **10**, 1988.

Poggio, T., K.R.K. Nielsen and H.K. Nishihara. "Zero-Crossings and Spatiotemporal Interpolation in Vision: Aliasing and Electrical Coupling Between Sensors," *Artificial Intelligence Laboratory Memo 675*, Massachusetts Institute of Technology, Cambridge, MA, 1982.

Poggio, G. and T. Poggio. "The analysis of stereopsis," *Ann. Rev. Neurosci.*, **7**, 379-412, 1984.

Poggio, T. "Early Vision: From Computational Structure to Algorithms and Parallel Hardware," *Computer Vision, Graphics, and Image Processing*, **31**, 1985.

Poggio, T. "Integrating Vision Modules with Coupled MRFs," *Artificial Intelligence Laboratory Working Paper 285*, Massachusetts Institute of Technology, Cambridge, MA, 1985.

Poggio, T. and staff. "MIT Progress in Understanding Images," *Proc. Image Understanding Workshop*, L. Bauman (ed.), Scientific Applications International Corp., McLean, VA, 1985.

Poggio T. and staff. "MIT Progress in Understanding Images," *Proc. Image Understanding Workshop*, L. Bauman (ed.), Scientific Applications International Corp., McLean, VA, 1987.

Poggio, T., H.L. Voorhees and A.L. Yuille. "Regularizing Edge Detection," *Artificial Intelligence Laboratory Memo 776*, Massachusetts Institute of Technology, Cambridge, MA, 1984.

Poggio, T., V. Torre, and C. Koch. "Computational Vision and Regularization Theory," *Nature*, **317**, 314-319, 1985.

Richards, W., and D.D. Hoffman. "Codon Constraints on Closed 2D Shapes," *Computer Vision, Graphics, and Image Processing*, **32**, 265-281, 1985.

Reichardt W. and T. Poggio. "Visual Control of Orientation in the Fly: I. A Quantitative Anaylsis," *Quart. Rev. Biophysics*, **3**, 311-375, 1976.

Reichardt W. and T. Poggio. "Visual Control of Orientation in the Fly: II. Towards the Underlying Neural Interactions," *Quart. Rev. Biophysics*, **9**, 377-439, 1976.

Rock, I. **Orientation and Form**, Academic Press, New York, 1973.

Terzopoulos, D. "Integrating Visual Information From Multiple Sources," In: **From Pixels to Predicates**, A.P. Pentland (ed.), Ablex Publishing Corp., Norwood, NJ, 1986.

Tikhonov A.N. and V.Y. Arsenin. **Solution of Ill-Pos Problems**, Winston and Wiley Publishers, Washington, DC, 1977.

Torre, V. and T. Poggio. "On Edge Detection," *Artificial Intelligence Laboratory Memo 768*, Massachusetts Institute of Technology, Cambridge, MA, 1984.

Ullman S. **The Interpretation of Visual Motion**, The MIT Press, Cambridge, MA, 1979.

Ullman, S. "Visual Routines," *Cognition*, **18**, 1984.

Verri, A. and T. Poggio. "Motion Field and Optical Flow: Qualitative Properties," *Artificial Intelligence Laboratory Memo 917*, Massachusetts Institute of Technology, Cambridge, MA, 1986.

Voorhees, H.L. and T. Poggio. "Detecting Textons and Texture Boundaries in Natural Images," *Proc Intl. Conf. on Computer Vision*, Computer Society of the IEEE, Washington, DC, 1987.

Waxman, A. "Image Flow Theory: A Framework for 3-D Inference from Time-Varying Imagery," In: **Advances in Computer Vision**, C. Brown (ed.), Lawrence Erlbaum Assocs, NJ, 1987.

Wyllie, J.C. "The Complexity of Parallel Computations," *Technical Report 79-387*, Dept. of Computer Science, Cornell University, Ithaca, NY, 1979

# Kalman Filter-based Algorithms for Estimating Depth from Image Sequences[1]

**Larry Matthies, Richard Szeliski, and Takeo Kanade**
Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

## Abstract

Using known camera motion to estimate depth from image sequences is an important problem in robot vision. Many applications of depth from motion, including navigation and manipulation, require algorithms that can estimate depth in an on-line, incremental fashion. This requires a representation that records the uncertainty in depth estimates and a mechanism that integrates new measurements with existing depth estimates to reduce the uncertainty over time. Kalman filtering provides this mechanism. Previous applications of Kalman filtering to depth from motion have been limited to estimating depth at the location of a sparse set of features. In this paper, we introduce a new, pixel-based (*iconic*) algorithm that estimates depth and depth uncertainty at each pixel and incrementally refines these estimates over time. We describe the algorithm for translations parallel to the image plane and contrast its formulation and performance to that of a feature-based Kalman filtering algorithm. We compare the performance of the two approaches by analyzing their theoretical convergence rates, by conducting quantitative experiments with images of a flat poster, and by conducting qualitative experiments with images of a realistic outdoor scene model. The results show that the new method is an effective way to extract depth from lateral camera translations and suggest that it will play an important role in low-level vision.

## 1 Introduction

Using known camera motion to estimate depth from image sequences is important in many applications of computer vision to robot navigation and manipulation. In these applications, depth from motion can be used by itself, as part of a multi-modal sensing strategy, or as a way to guide stereo matching. Many applications require a depth estimation algorithm that operates in an on-line, incremental fashion. Such algorithms require a depth representation that includes not only the current depth estimate, but also an estimate of the uncertainty in the current depth estimate.

Previous work [Baker87] [Broida86] [Faugeras86] [Hallam83] [Matthies87b] [Matthies87c] [Rives86] has identified Kalman filtering as a viable framework for this problem, because it incorporates representations of uncertainty and provides a mechanism for incrementally reducing uncertainty over time. To date, applications of this framework have largely been restricted to estimating the positions of a sparse set of trackable features, such as points or line segments. While this is adequate for many robotics applications, it requires reliable feature extraction and it fails to describe large areas of the image. Another line of work has addressed the problem of extracting dense depth or displacement estimates from image sequences. However, these approaches either have been restricted to two frame analysis [Anandan85] or have used batch processing of the image sequence, for example via spatio-temporal filtering [Heeger87].

In this paper we introduce a new, pixel-based (*iconic*) approach to incremental depth estimation and compare it mathematically and experimentally to a feature-based approach we developed previously [Matthies87b]. The new approach represents depth and depth variance at every pixel and uses Kalman filtering to extrapolate and update the pixel-based depth representation. The algorithm uses correlation to measure optical flow and to estimate the variance in the flow, then uses the known camera motion to convert the flow field into a depth map. It then uses the Kalman filter to generate an updated depth map from a weighted combination of the new measurements and the prior depth estimates. Regularization is employed to smooth the depth map and to fill in underconstrained areas. The resulting algorithm is parallel, uniform, and can take advantage of mesh-connected or multi-resolution (pyramidal) processing architectures.

The remainder of this paper is structured as follows. In the next section, we give a brief review of Kalman filtering and introduce our overall approach to Kalman filtering of depth. We then describe our new, pixel-based depth from motion algorithm and the feature-based algorithm to which it will be compared. We then analyze the theoretical accuracy of both methods, compare them both to the theoretical accuracy of stereo matching, and verify this analysis experimentally using images of a flat scene. We also show the performance of both methods on images of realistic outdoor scene models. In the final section, we discuss the promise and the problems involved in extending the new method to arbitrary motion. We also conclude that the ideas and results presented apply directly to the much broader problem of integrating depth information from multiple sources.

[1]This research was sponsored in part by DARPA, monitored by the Air Force Avionics Lab under contract F33615-87-C-1499 and in part by a postgraduate fellowship from the FMC Corporation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies.

| | |
|---|---|
| **Models** | |
| system model | $u_t = \Phi_t u_{t-1} + \eta_t, \eta_t \sim N(0, Q_t)$ |
| measurement model | $d_t = H_t u_t + \xi_t, \ \xi_t \sim N(0, R_t)$ |
| prior model | $\mathrm{E}[u_0] = \hat{u}_0, \ \mathrm{Cov}[u_0] = P_0$ |
| (other assumptions) | $\mathrm{E}[\eta_t \xi_j^T] = 0$ |
| **Prediction phase** | |
| state estimate extrapolation | $\hat{u}_t^- = \Phi_{t-1} \hat{u}_{t-1}^+$ |
| state covariance extrapolation | $P_t^- = \Phi_{t-1} P_{t-1}^+ \Phi_{t-1}^T + Q_{t-1}$ |
| **Update phase** | |
| state estimate update | $\hat{u}_t^+ = \hat{u}_t^- + K_t[d_t - H_t \hat{u}_t^-]$ |
| state covariance update | $P_t^+ = [I - K_t H_t] P_t^-$ |
| Kalman gain matrix | $K_t = P_t^- H_t^T [H_t P_t^- H_t^T + R_t]^{-1}$ |

Table 1: Kalman filter equations

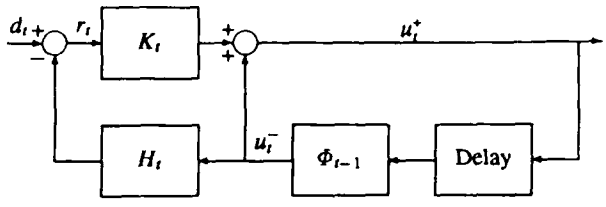



Figure 1: Kalman filter block diagram

## 2 Estimation framework

The depth from motion algorithms described in this paper use a sequence of images taken with small inter-frame displacements [Bolles87]. The advantage of using such a sequence is that the correspondence problem between two successive images is reduced. The disadvantage is that the individual depth measurements are less precise because of the very small baselines involved. To overcome this latter problem, information from each pair of frames must be integrated over time. For many robotics applications it is desirable to process these images using a real-time rather than batch process, with an updated depth estimate being generated after each new image is acquired. The incremental algorithm also has the advantage of requiring less storage, since only the current estimate and its uncertainty model are required.

A powerful technique for doing real-time estimation of such dynamic systems is the Kalman filter. This formulation allows for the integration of information over time, and is robust with respect to both system and sensor noise. The filter is based on three separate probabilistic models, as shown in Table 1. The first model, the *system model*, describes the evolution over time of the current state vector $u_t$. The transition between states is characterized by the known transition matrix $\Phi_t$ and the addition of Gaussian noise with a covariance $Q_t$. The second model, the *measurement* (or *sensor*) *model*, relates the measurement vector $d_t$ to the current state through a measurement matrix $H_t$ and the addition of Gaussian noise with a covariance $R_t$. The third model, the *prior model*, describes the knowledge about the system state $\hat{u}_0$ and its covariance $P_0$ before the first measurement is taken. The sensor and process noise are assumed to be uncorrelated.

The Kalman filter algorithm operates in two phases: extrapolation (prediction) and update (correction). The previous state estimate $\hat{u}_{t-1}^+$ is used to predict the current state $\hat{u}_t^-$. At the same time, the previous state covariance $P_{t-1}^+$ is extrapolated to the predicted state covariance $P_t^-$. This predicted covariance is used to compute the new Kalman gain matrix $K_t$ and the updated covariance matrix $P_t^+$. Finally, the measurement residual $d_t - H_t \hat{u}_t^-$ is weighted by the gain matrix $K_t$ and added to the predicted state $u_t^-$ to yield the updated state $u_t^+$.

A block diagram for the Kalman filter is given in Figure 1.

Kalman filtering is usually applied to systems with fairly small numbers of state variables. In the domain of motion sequence analysis, it has been used to track sparse features [Matthies87c], but has not previously been used in conjunction with *dense* fields such as iconic depth maps. We will briefly describe how this estimation framework is instantiated in our depth from motion algorithm. Section 3 describes the details of the implementation for lateral camera motion; extensions to general motion are considered in [Matthies87d].

The first step in designing a Kalman filter is to specify a representation for the state vector. In our case, this is a representation of the depth at every pixel $(x, y)$ in the current image. We choose to represent the inverse depth $u(x, y) = 1/Z(x, y)$, which we call "disparity", plus the variance in the disparity, $\sigma_d^2(x, y)$. There are several reasons for representing disparity instead of the actual depth $Z(x, y)$. For example, disparity can be linearly related to optical flow measurements, it is better conditioned for distant objects, and, for lateral camera motion, the scaled disparity and variance can be used directly to set search limits on correspondence in the subsequent image.

The system model uses the current depth map and an estimate of the camera motion to predict a depth map for the next image in the sequence. This is implemented by using the predicted optical flow to warp the depth map, then resampling the warped map to compute the predicted disparity at each pixel. The measurement model simply produces a measurement of the disparity at every pixel, so that $H_t = I$. The disparity measurement in turn is based on an optical flow measurement obtained by a correlation-based flow estimator. Estimates of the variance $\sigma_d^2(x, y)$ of disparity measurements are computed from the variance of the optical flow. These variance estimates are essential, because they characterize the difference between reliable measurements, such as those obtained in highly textured areas or near strong edges, and unreliable measurements, such those obtained in areas of uniform intensity.

Finally, the prior model embeds prior knowledge about the scene. In particular, smoothness constraints, which require nearby points to have similar disparity, can be modeled by off-diagonal elements in the prior covariance matrix $P_0$. This is equivalent to modeling the disparity map as a Markov Random Field [Szeliski87a].

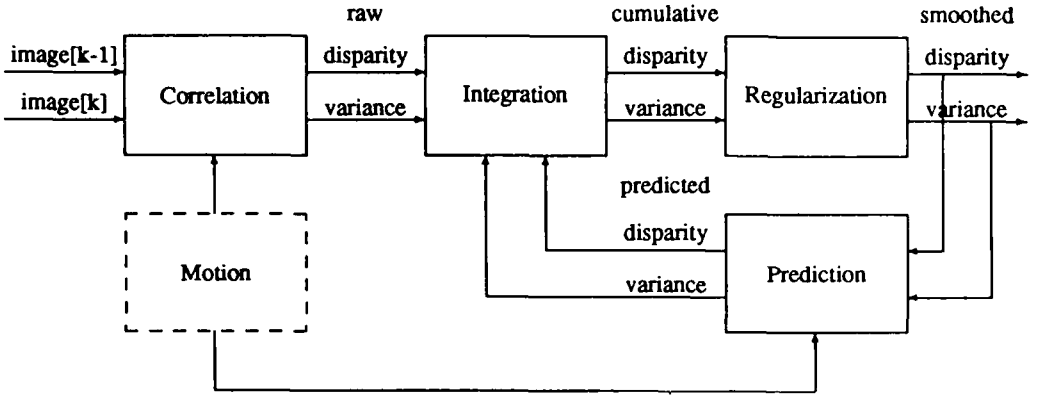



Figure 2: Iconic depth estimation block diagram

# 3 Iconic depth estimation

Our implementation of this framework consists of four main stages (see Figure 2). The first stage uses correlation to produce a measurement of the disparity at each pixel and an estimate of its associated variance. The second stage integrates this information with the disparity map predicted from the previous time step. The third stage uses regularization-based smoothing to reduce measurement noise and to fill in areas of unknown disparity. The last stage uses known camera motion to predict the disparity field that will be seen in the next frame and re-samples the field to keep it pixel based. Here we deal only with camera translation parallel to the image plane; in this case, estimating disparity is equivalent to estimating optical flow. Extensions to arbitrary camera motion are described in [Matthies87d].

## 3.1 Measurement (correlation)

The problem of extracting optical flow from a sequence of intensity images has been extensively studied in computer vision. Early approaches used the ratio of the spatial and temporal image derivatives [Horn81], while more recent approaches have used correlation between images [Anandan85] or spatio-temporal filtering [Heeger87]. The approach used in this paper is a simple version of correlation-based matching. This technique, which has been called the Sum of Squared Differences (SSD) method [Anandan85], integrates the squared intensity difference between two shifted images over a small area. For the case of lateral motion, this error measure is

$$e_t(d;x,y) = \int\int w(\lambda,\eta)[f_t(x-d+\lambda,y+\eta)-f_{t-1}(x+\lambda,y+\eta)]^2\,d\lambda\,d\eta,$$

where $f_t$ and $f_{t-1}$ are the two intensity images and $w(\lambda,\eta)$ is a weighting function. This measure is computed at each pixel for a number of possible disparity values $d$. In Anandan's algorithm, a coarse-to-fine technique is used to limit the range of possible flow values. In our images, the possible range of values is small (since we are using small-motion sequences), so a single-resolution algorithm is used[2]. The resulting error surface $e(d;x,y)$ is approximately parabolic around the minimum. The flow value with the lowest error is taken as the estimator output $d(x,y)$ and the second derivative of the error surface is used to compute its variance.

The implementation of the SSD algorithm is particularly simple and efficient for flow parallel to the image raster. Each scanline of the two image frames is first magnified by a factor of 4 by cubic interpolation. The images are then shifted using sub-pixel displacements $d_k$ and the SSD measure $e_k$ is computed using a 5 × 5-pixel square window. The minimum error $(d_{\hat{k}}, e_{\hat{k}})$ is found and a parabola

$$e(d) = ad^2 + bd + c$$

is fit to this point and its two neighbors $(d_{\hat{k}-1}, e_{\hat{k}-1})$ and $(d_{\hat{k}+1}, e_{\hat{k}+1})$. The minimum of this parabola establishes the flow estimate (to sub-sub-pixel precision). Appendix A shows that the variance of the flow measurement is

$$\mathrm{Var}(d) = 2\sigma^2/a,$$

where $\sigma^2$ is the variance of the image noise process. (see [Matthies87d] for a derivation). The appendix also shows that adjacent flow estimates are correlated over both space and time; the significance of this fact will be considered in Section 5.1.

The raw flow and variance estimates are scaled to units of inverse depth using knowledge of the camera motion and the calibration parameters of a pin-hole camera model. This facilitates the integration of information when the camera motion is not linear, e.g. for widening baseline stereo [Xu85] or for orthogonal camera motions (Section 5.2). For arbitrary camera motion, the uncertainty in two-dimensional flow can be described by covariance matrices obtained from an error analysis similar to the one above [Anandan85]. Estimates of

[2] It may be necessary to use a larger search range at first, but once the estimator has "latched on" to a good disparity map, the predicted disparity and disparity variance can be used to further limit the search.

the disparity and disparity variance can be obtained from such flow measurements via the equations of motion [Matthies87d].

## 3.2 Update (integration)

The next stage in the iconic depth estimator is the integration of the new disparity measurements with the predicted disparity map (this step is omitted for the first pair of images). For now, we will assume that each value in the measured or predicted disparity map is not correlated with its neighbors, so that the map updating can be done at each pixel independently. The extension of this model to account for the correlated nature of disparity maps is discussed later.

To update a pixel value, we first compute the variance of the updated disparity estimate

$$p_t^+ = ((p_t^-)^{-1} + (\sigma_d^2)^{-1})^{-1} = \frac{p_t^- \sigma_d^2}{p_t^- + \sigma_d^2}$$

and the Kalman filter gain $K$

$$K = \frac{p_t^+}{\sigma_d^2} = \frac{p_t^-}{p_t^- + \sigma_m^2}.$$

We then update the disparity value by using the Kalman filter update equation

$$u_t^+ = u_t^- + K(d - u_t^-)$$

where $u_t^-$ and $u_t^+$ are the predicted and updated disparity estimates and $d$ is the new disparity measurement. This update equation can also be written as

$$u_t^+ = p_t^+ \left( \frac{u_t^-}{p_t^-} + \frac{d}{\sigma_d^2} \right).$$

The latter form shows that the updated disparity estimate is a linear combination of the predicted and measured values, inversely weighted by their respective variances.

## 3.3 Smoothing (regularization)

The raw depth or disparity values obtained from optical flow measurements can be very noisy, especially in areas of uniform intensity. We employ smoothness constraints to reduce the noise and to "fill in" underconstrained areas. The earliest example of this approach is that of Horn and Schunck [Horn81], who smoothed optical flow fields by jointly minimizing the error in the flow equation and the departure of the flow field from smoothness. More recently, this approach has been formalized using the theory of regularization [Terzopoulos86a] and extended to use two-dimensional confidence measures equivalent to local covariance estimates [Anandan85],[Nagel86].

For our application, smoothing is done on the disparity field, using the inverse variance of the disparity estimate as the confidence in each measurement. The smoother we use is the generalized piecewise continuous spline under tension [Terzopoulos86b], which uses finite element relaxation to compute the smoothed field. The algorithm is implemented with a three-level coarse-to-fine strategy to speed convergence and is amenable to implementation on a parallel computer.

The smoothing stage can be viewed as the part of the Kalman filtering algorithm that incorporates prior knowledge about the smoothness of the disparity map. As shown in [Szeliski87a], a regularization-based smoother is equivalent to a prior model with a correlation function defined by the degree of the stabilizing spline (e.g. membrane or thin plate). The resulting prior covariance matrix contains off-diagonal elements modeling the covariance of neighboring pixels. An optimal implementation of the Kalman filter would require carrying this entire covariance matrix, with non-zero correlations between the depths at neighboring pixels, through the update and prediction stages. This would significantly complicate our algorithm. Our choice to explicitly model only the variance at each pixel, with covariance information implicitly modeled in a fixed regularization stage, has worked well in practice.

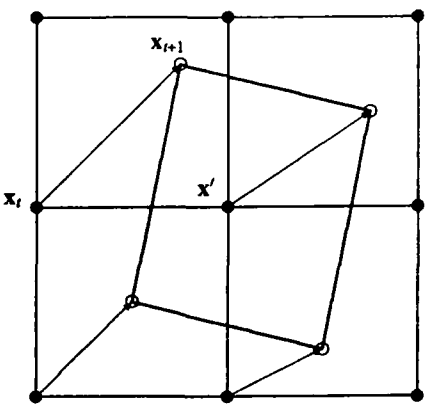



Figure 3: Illustration of disparity prediction stage

## 3.4 Prediction (warping and resampling)

The prediction stage of the Kalman filter must predict both the depth and the depth uncertainty for each pixel in the next image. We will describe the disparity extrapolation first, then consider the uncertainty extrapolation.

Lateral camera translation shifts each point $\mathbf{x}_t = (x, y)$ in the current image to the point $\mathbf{x}_{t+1}$ in the next image. This shift can be viewed as warping the disparity map (Figure 3). The shift is computed by using the perspective projection equations

$$x_t = \frac{X}{Z}, \quad y_t = \frac{Y}{Z}, \quad u_t = \frac{1}{Z}$$

and the known lateral translation $T_x$ to obtain

$$x_{t+1} = x_t - T_x u_t, \quad y_{t+1} = y_t, \quad u_{t+1} = u_t.$$

In general this prediction process will yield estimates of disparity in between pixels in the new image (Figure 3), so we need to resample to obtain predicted disparity at pixel locations. For a given pixel $\mathbf{x}'$ in the new image, we find the square of extrapolated pixels that overlap $\mathbf{x}'$ and compute the disparity at $\mathbf{x}'$ by bi-linear interpolation of the extrapolated disparities. Note that it may be possible to detect occlusions

by recording where the extrapolated squares turn away from the camera. Detecting "disocclusions", where newly visible areas become exposed, is not possible if the disparity field is assumed to be continuous, but is possible if disparity discontinuities have been detected.

Uncertainty will increase in the prediction phase due to errors from many sources, including uncertainty in the motion parameters, errors in calibration, and inaccurate models of the camera optics. A simple approach to modeling these errors is to lump them together by inflating the current variance estimates by a small multiplicative factor in the prediction stage,

$$p_{t+1}^{-} = (1+\epsilon)p_t^{+}. \tag{1}$$

In the Kalman filtering literature this is known as exponential age-weighting of measurements [Maybeck79], because it decreases the weight given to previous measurements by an exponential function of time. This is the approach used in our implementation. We first inflate the variance in the current disparity map using equation (1), then warp and interpolate the variance map in the same way as the disparity map. A more exact approach is to attempt to model the individual sources of error and to propagate their effects through the prediction equations (see [Matthies87d], Appendix C).

# 4 Feature based depth estimation

The dense, iconic depth estimation algorithm described in the previous section can be compared with existing depth estimation methods based on sparse feature tracking. Such methods [Ayache87] [Broida86] [Hallam83] [Matthies87b] typically define the state vector to be the parameters of the 3-D object being tracked, which is usually a point or straight line segment. The 3-D motion of the object between frames defines the system model of the filter and the perspective projection of the object onto each image defines the measurement model. This implies that the measurement equations (the perspective projection) are non-linear functions of the state variables (e.g. the 3-D position vector); this requires linearization in the update equations and implies that the error distribution of the 3-D coordinates will *not* be Gaussian. In the case of arbitrary camera motion, a further complication is that it is difficult to reliably track features between frames. In this section, we will describe an approach to feature-based Kalman filtering for lateral motion that tracks edgels along each scanline and avoids the problems associated with non-linear measurement equations. Extensions to arbitrary motion can be based on the method presented here.

## 4.1 Kalman filter formulation for lateral motion

For lateral camera motion, the position of a feature on a scanline is a linear function of the distance moved by the camera, since

$$\Delta x = T_x d \quad \Leftrightarrow \quad x_t = x_0 + tT_x d$$

where $x_0$ is the position of the feature in the first frame and $d$ is the inverse depth of the feature. The *epipolar plane image* method [Bolles87] exploits these characteristics by extracting lines in "space-time" (epipolar plane) images formed by concatenating scanlines from an entire image sequence. However, sequential estimation techniques like Kalman filtering are a more practical approach to this problem because they allow images to be processed on-line by incrementally refining the depth model [Baker87] [Matthies87b].

Taking $x_0$ and $d$ as the state variables defining the location of the feature, instead of the 3-D coordinates $X$ and $Z$, keeps the entire estimation problem linear. This is advantageous because it avoids the approximations needed for error estimation with non-linear equations. For point features, if the position of the feature in each image is given by the sequence of measurements $\tilde{\mathbf{x}} = [\tilde{x}_0, \tilde{x}_1, \ldots \tilde{x}_n]^T$, knowledge of the camera position for each image allows the feature location to be determined by fitting a line to the measurement vector $\tilde{\mathbf{x}}$:

$$\tilde{\mathbf{x}} = \mathbf{H}\begin{bmatrix} x_0 \\ d \end{bmatrix} \tag{2}$$

where $\mathbf{H}$ is a $(2 \times n+1)$ matrix whose first column contains all 1's and whose second column is defined by the camera position for each frame, relative to the initial camera position. This fit can be computed sequentially by accumulating the terms of the normal equation solution for $x_0$ and $d$. The covariance matrix $\Sigma$ of $x_0$ and $d$ can be determined from the covariance matrix of the measurement vector $\tilde{\mathbf{x}}$.

The approach outlined above uses the position of the feature in the first frame $x_0$ as one of the two state variables. We can reformulate this in terms of the current frame by taking $x_t$ and $d$ to be the state variables. Assuming that the camera motion is exact and that measured feature positions have normally distributed uncertainty with variance $\sigma_e^2$, the initial state vector and covariance matrix are expressed in terms of image coordinates as

$$\begin{aligned} x_1 &= \tilde{x}_1 \\ d &= \frac{\tilde{x}_1 - \tilde{x}_0}{T_1} \\ P_0^{+} &= \sigma_e^2 \begin{bmatrix} 1 & 1/T_1 \\ 1/T_1 & 2/T_1^2 \end{bmatrix} \end{aligned}$$

where $T_1$ is the camera translation between the first and second frame. The covariance matrix comes from applying standard linear error propagation methods to the equations for $x_1$ and $d$ [Maybeck79].

After initialization, if $T_t$ is the translation between frames $t-1$ and $t$, the motion equations that transform the state vector and covariance matrix to the current frame are

$$u_t^{-} = \begin{bmatrix} x_t^{-} \\ d_t^{-} \end{bmatrix} = \begin{bmatrix} 1 & T_t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1}^{+} \\ d_{t-1}^{+} \end{bmatrix} = \Phi_t u_{t-1}^{+} \tag{3}$$

$$P_t^{-} = \Phi_t P_{t-1}^{+} \Phi_t^T. \tag{4}$$

The superscript minuses indicate that these estimates do not incorporate the measured edge position at time $t$. The newly measured edge position $\tilde{x}_t$ is incorporated by computing the updated covariance matrix $P_t^{+}$, a gain matrix $K$, and the updated parameter vector $u_t^{+}$:

$$P_t^{+} = \{(P_t^{-})^{-1} + S\}^{-1} \quad \text{where} \quad S = \frac{1}{\sigma_e^2}\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$K = \frac{1}{\sigma_e^2} P_t^+ \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$u_t^+ = u_t^- + K[\bar{x}_t - x_t^-].$$

Since these equations are linear, we can see how uncertainty decreases as the number of measurements increases by computing the sequence of covariance matrices $P_t$, given only the measurement uncertainty $\sigma_e^2$ and the sequence of camera motions $T_t$. This is addressed in Section 5.1.

Note that the equations above can be generalized to arbitrary, uncertain camera motion using either the $x, y, d$ image-based parameterization of point locations or an $X, Y, Z$ three-dimensional parameterization. The resulting estimation problem will be nonlinear.

## 4.2 Feature extraction and matching

To implement the feature-based depth estimator, we must specificy how to extract feature positions, how to estimate the noise level in those positions, and how to track features from frame to frame. For lateral motion, with image flow parallel to the scanlines, tracking edgels on each scanline is a natural implementation. Therefore, in this section we will describe how we extract edges to sub-pixel precision, how we estimate the variance of the edge positions, and how we track edges from frame to frame.

For one-dimensional signals, estimating the variance of edge positions has been addressed in [Canny86]. We will review this analysis before considering the general case. In one dimension, edge extraction amounts to finding the zero crossings in the second derivative of the Gaussian-smoothed signal, which is equivalent to finding zero-crossings after convolving the image with a second derivative of Gaussian operator,

$$F(x) = \frac{d^2 G(x)}{dx^2} * I(x).$$

We assume that the image $I$ is corrupted by white noise with variance $\sigma_n^2$. Splitting the response of the operator into that due to the signal, $F_s$, and that due to noise, $F_n$, edges are marked where

$$F_s(x) + F_n(x) = 0. \tag{5}$$

An expression for the edge variance is obtained by taking a first-order Taylor expansion of the deterministic part of the response in the vicinity of the zero crossing, then taking mean square values. Thus, if the zero crossing occurs at $x_0$ in the noise free signal and $x_0 + \delta x$ in the noisy signal, we have

$$F(x_0 + \delta x) \approx F_s(x_0) + F_s'(x_0)\delta x + F_n(x_0 + \delta x) = 0, \tag{6}$$

so that

$$\delta x = \frac{-(F_n(x_0 + \delta x) + F_s(x_0))}{F_s'(x_0)}. \tag{7}$$

The presence of a zero crossing implies that $F_s(x_0) = 0$ and the assumption of zero mean noise implies that $E[F_n(x_0)] = 0$. Therefore, the variance of the edge position is

$$E[\delta x^2] = \sigma_e^2 = \frac{\sigma_n^2 E[(F_n(x_0))^2]}{(F_s'(x_0))^2}. \tag{8}$$

In a discrete implementation, $E[(F_n(x_0))^2]$ is the sum of the squares of the coefficients in the convolution mask. $F_s'(x_0)$ is the slope of the zero crossing and is approximated by fitting a local curve to the filtered image. The zero crossing of this curve gives the estimate of the sub-pixel edge position.

For two-dimensional images, an analogous edge operator is a directional derivative filter with a derivative of Gaussian profile in one direction and a Gaussian profile in the orthogonal direction. Assuming that the operator is oriented to take the derivative in the direction of the gradient, the analysis above will give the variance of the edge position in the direction of the gradient (see [Nalwa86] for an alternate approach). However, for edge tracking along scanlines, we require the variance of the edge position in the scanline direction, not the gradient direction. This is straightforward to compute for the difference of Gaussian (*DOG*) edge operator; the required variance estimate comes directly from equations (5) - (8), replacing $F$ with the *DOG* and $F'$ with the partial derivative $\partial/\partial x$. Details of the discrete implementation in this case are similar to those described above. Experimentally, the cameras and digitizing hardware we use provide 8-bit images with intensity variance $\sigma_n^2 \approx 4$.

It is worth emphasizing that estimating the variance of edge positions is more than a mathematical nicety; it is valuable in practice. The uncertainty in the position of an edge is affected by the contrast of the edge, the amount of noise in the image, and, in matching applications such as this one, by the edge orientation. For example, in tracking edges under lateral motion, edges that are close to horizontal provide much less precise depth estimates than edges that are vertical. Estimating variance quantifies these differences in precision. Such quantification is important in predictive tracking, fitting surface models, and applications of depth from motion to constraining stereo. These remarks of course apply to image features in general, not just to edges.

Tracking features from frame to frame is very simple if either the camera motion is very small or the feature depth is already known quite accurately. In the former case, a search window is defined that limits the feature displacement to a small number of pixels from the position in the previous image. For the experiments described in Section 5, tracking was implemented this way, with a window width of two pixels. Alternatively, when the depth of a feature is already known fairly accurately, the position of the feature in a new image can be predicted from equation (3) to be

$$x_t^- = x_{t-1}^+ + T_t d_{t-1}^+,$$

the variance of the prediction can be determined from equation (4), and a search window can be defined as a confidence interval estimated from this variance. This allows tight search windows to be defined for existing features even when the camera motion is not small. A simplified version of this procedure is used in our implementation to ensure that candidate edge matches are consistent with the existing depth model. The predefined search window is scanned for possible matches, and these are accepted only if they lie within some distance of the predicted edge location. Additional acceptance criteria

require the candidate match to have properties similar to those of the feature in the previous image; for edges, these properties are edge orientation and edge strength (gradient magnitude or zero-crossing slope). Given knowledge of the noise level in the image, this comparison function can be defined probabilistically as well, but we have not pursued this direction.

Finally, if the noise level in the image is unknown it can be estimated from the residuals of the observations after $x$ and $d$ have been determined. Such methods are discussed in [Mikhail76] for batch oriented techniques analogous to equation (2) and in [Maybeck82] for Kalman filtering.

# 5 Evaluation

In this section, we compare the performance of the iconic and feature-based depth estimation algorithms in three ways. First, we perform a mathematical analysis of the reduction in depth variance as a function of time. Second, we use a sequence of images of a flat scene to determine the quantitative performance of the two approaches and to check the validity of our analysis. Third, we test our algorithms on images of realistic scenes with complicated variations in depth.

## 5.1 Mathematical analysis

We wish to compare the theoretical variance of the depth estimates obtained by the iconic method of Section 3 to those obtained by the feature-based method of Section 4. We will also compare the accuracy of both methods to the accuracy of stereo matching with the first and last frames of the image sequence. To do this, we will derive expressions for the depth variance as a function of the number of frames processed, assuming a constant noise level in the images and constant camera motion between frames. For clarity, we will assume this motion is $T_x = 1$.

### Iconic approach

For the iconic method, we will ignore process noise in the system model and assume that the variance of successive flow measurements is constant. For lateral motion, the equations developed in Section 2 can be simplified to show that the Kalman filter simply computes the average flow [Gelb74]. Therefore, a sequence of flow measurements $\Delta x_1, \Delta x_2, \ldots, \Delta x_t$ is equivalent to the following batch measurement equation

$$\Delta \mathbf{x} = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \cdot \\ \cdot \\ \cdot \\ \Delta x_t \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix} d = \mathbf{H}d.$$

Estimating $d$ by averaging the flow measurements implies that

$$d = \frac{1}{t} H^T \Delta \mathbf{x} = \frac{1}{t} \sum_{i=1}^{t} \Delta x_i. \tag{9}$$

If the flow measurements were independent with variance $2\sigma_n^2/a$, where $\sigma_n$ is the noise level in the image (Appendix A), the resulting variance of the disparity estimate would be

$$\frac{2\sigma_n^2}{ta}. \tag{10}$$

However, the flow measurements are not actually independent. Because noise is present in every image, flow measurements between frames $i-1$ and $i$ will be correlated with measurements for frames $i$ and $i+1$. Appendix A shows that a sequence of correlation-based flow measurements that track the same point in the image sequence will have the following covariance matrix:

$$P_m = \frac{\sigma_n^2}{a} \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & & & & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix}$$

where $\sigma^2$ is the level of noise in the image and $a$ reflects the local slope of the intensity surface. With this covariance matrix, averaging the flow measurements actually yields the following variance for the estimated flow:

$$\sigma_f^2(t) = \frac{1}{t^2} H^T P_m H = \frac{2\sigma^2}{t^2 a}. \tag{11}$$

This is interesting and rather surprising. Comparing equations (10) and (11), the correlation structure that exists in the measurements means that the algorithm converges faster than we first expected.

With correlated measurements, averaging the flow measurements in fact is a sub-optimal estimator for $d$. The optimal estimator is obtained by substituting the expressions for $H$ and $P_m$ into the batch solution equations [Maybeck79]

$$d = (H^T P_m^{-1} H)^{-1} H^T P_m^{-1} \Delta \mathbf{x} \tag{12}$$

and

$$\sigma_d^2 = (H^T P_m^{-1} H)^{-1}. \tag{13}$$

This estimator does not give equal weight to all flow measurements; instead, measurements near the center of the sequence receive more weight than those near the end. The variance of the depth estimate is

$$\sigma_f^2(t) = \frac{12\sigma_n^2}{t(t+1)(t+2)a}.$$

The optimal convergence is cubic, whereas the convergence of the averaging method we implemented is quadratic. Developing an incremental version of the optimal estimator requires extending our Kalman filter formulation to model the correlated nature of the measurements. This extension is currently being investigated.

### Feature-based approach

For the feature based approach, the desired variance estimates come from computing the sequence of covariance matrices $P_t$, as mentioned at the end of Section 4.1. A closed form expression for this matrix is easier to obtain from the batch method suggested by equation (2) than from the Kalman filter formulation and yields an equivalent result. Taking the constant camera translation to be $T_x = 1$ for simplicity, equation

(2) expands to

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x}_0 \\ \tilde{x}_1 \\ \cdot \\ \cdot \\ \cdot \\ \tilde{x}_t \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ 1 & t \end{bmatrix} \begin{bmatrix} x_0 \\ d \end{bmatrix} = H\mathbf{u}. \qquad (14)$$

Recall that $\tilde{x}_i$ are the edge positions in each frame, $x_0$ is the best fit edge position in the first frame, and $d$ is the best fit displacement or flow between frames. Since we assume that the measured edge positions $\tilde{x}_i$ are independent with equal variance $\sigma_e^2$, we find that

$$P_F = \begin{bmatrix} \sigma_x^2 & \sigma_{xd} \\ \sigma_{xd} & \sigma_d^2 \end{bmatrix} = \sigma_e^2 \begin{bmatrix} \sum_{i=0}^{t} 1 & \sum_{i=0}^{t} i \\ \sum_{i=0}^{t} i & \sum_{i=0}^{t} i^2 \end{bmatrix}^{-1}. \qquad (15)$$

The summations can be expressed in closed form, leading to the conclusion that

$$\sigma_F^2(t) = \frac{12\sigma_e^2}{t(t+1)(t+2)}. \qquad (16)$$

The variance of the displacement or flow estimate $d$ thus decreases as the cube of the number of images. This expression is identical in structure to the optimal estimate for the iconic approach, the only difference being the replacement of the variance of the SSD minimum by the variance of the edge position. Thus, if our estimators incorporate appropriate models of measurement noise, the iconic and feature-based methods theoretically achieve the same rate of convergence. This is surprising, given that the basic Kalman filter for the iconic method maintains only one state parameter ($d$) for each pixel, whereas the feature-based method maintains two per feature ($x_0$ and $d$). We suspect that an incremental version of the optimal iconic estimator will require the same amount of state as the feature-based method.

### Comparison with stereo

To compare these methods to stereo matching on the first and last frames of the image sequence, we must scale the stereo disparity and its uncertainty to be commensurate with the flow between frames. This implies dividing the stereo disparity by $t$ and its uncertainty by $t^2$. For the iconic method, we assume that the uncertainty in a stereo measurement will be the same as that for an individual flow measurement. Thus, the scaled uncertainty is

$$\sigma_{IS}^2(t) = \frac{2\sigma_n^2}{t^2 a}.$$

This is the same as is achieved by our incremental algorithm that processes all of the intermediate frames. Therefore, processing the intermediate frames as we do (that is, ignoring the temporal correlation of the measurements) may improve the reliability of the matching, but in this case it does not improve precision.

For the feature-based approach, the uncertainty in stereo disparity is twice the uncertainty $\sigma_e^2$ in the feature position; the scaled uncertainty is therefore

$$\sigma_{FS}^2(t) = \frac{2\sigma_e^2}{t^2}.$$

In this case using the intermediate frames helps, since

$$\frac{\sigma_F(t)}{\sigma_{FS}(t)} = \frac{1}{O(\sqrt{t})}.$$

Thus, extracting depth from a small-motion image sequence has several advantages over stereo matching between the first and last frames. The ease of matching is increased, reducing the number of correspondence errors. Occlusion is less of a problem, since it can be predicted from early measurements. Finally, better accuracy is available by using the feature based method or the optimal version of the iconic method.

## 5.2 Quantitative experiments: flat scenes

The goals of our quantitative evaluation were to examine the actual convergence rates of the depth estimators, to assess the validity of the noise models, and to compare the performance of the iconic and feature-based algorithms. To obtain ground truth depth data, we used the facilities of the Calibrated Imaging Lab at CMU to digitize a sequence of images of a flat-mounted poster. We used a Sony XC-37 CCD camera with a 16mm lens, which gave a field of view of 36 degrees. The poster was set about 20 inches (51 cm) from the camera. The camera motion between frames was 0.04 inches (1 mm), which gave an actual flow of approximately two pixels per frame in 480x512 images. For convenience, our experiments were run on images reduced to 240x256 by Gaussian convolution and subsampling. The image sequence we will discuss here was taken with vertical camera motion. This proved to give somewhat better results than horizontal motion; we attribute this to jitter in the scanline clock, which induces more noise in horizontal flow than in vertical flow.

Figure 4 shows the poster and the edges extracted from it. For both the iconic and the feature-based algorithms, a ground truth value for the depth was determined by fitting a plane to the measured values. The level of measurement noise was then estimated by computing the RMS deviation of the measurements from the plane fit. Optical aberrations made the flow measurements consistently smaller near the periphery of the image than the center, so the RMS calculation was performed over only the center quarter of the image. Note that all experiments described in this section did *not* use regularization to smooth the depth estimates, so the results show only the effect of the Kalman filtering algorithm.

To examine the convergence of the Kalman filter, the RMS depth error was computed for the iconic and the feature-based algorithms after processing each image in the sequence. We computed two sets of statistics, one for "sparse" depth and one for "dense" depth. The sparse statistic computes the RMS error for only those pixels where both algorithms gave depth estimates (that is, where edges were found), whereas the dense statistic computes the RMS error of the iconic algorithm over the full image. Figure 5 plots the relative RMS errors as a function of the number of images processed. Comparing the sparse error curves, the convergence rate of the iconic algorithm is slower than the feature-based algorithm, as expected. The relative heights of the two curves will depend on the relative sizes and shapes of the correlation window and the edge operator. In this particular experiment, both methods converged to

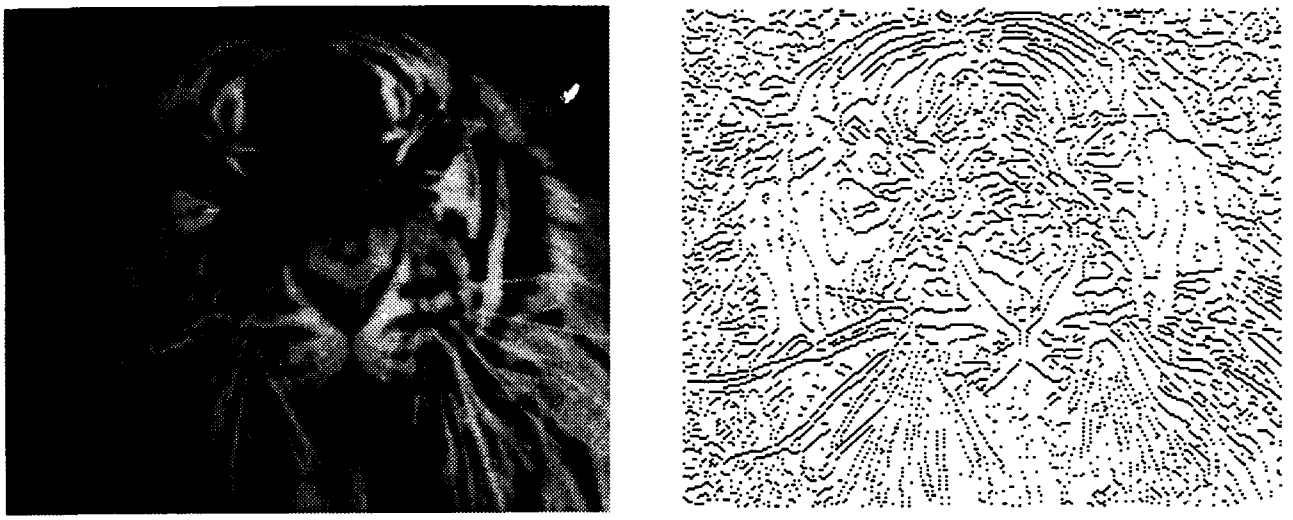


Figure 4: Tiger image and edges

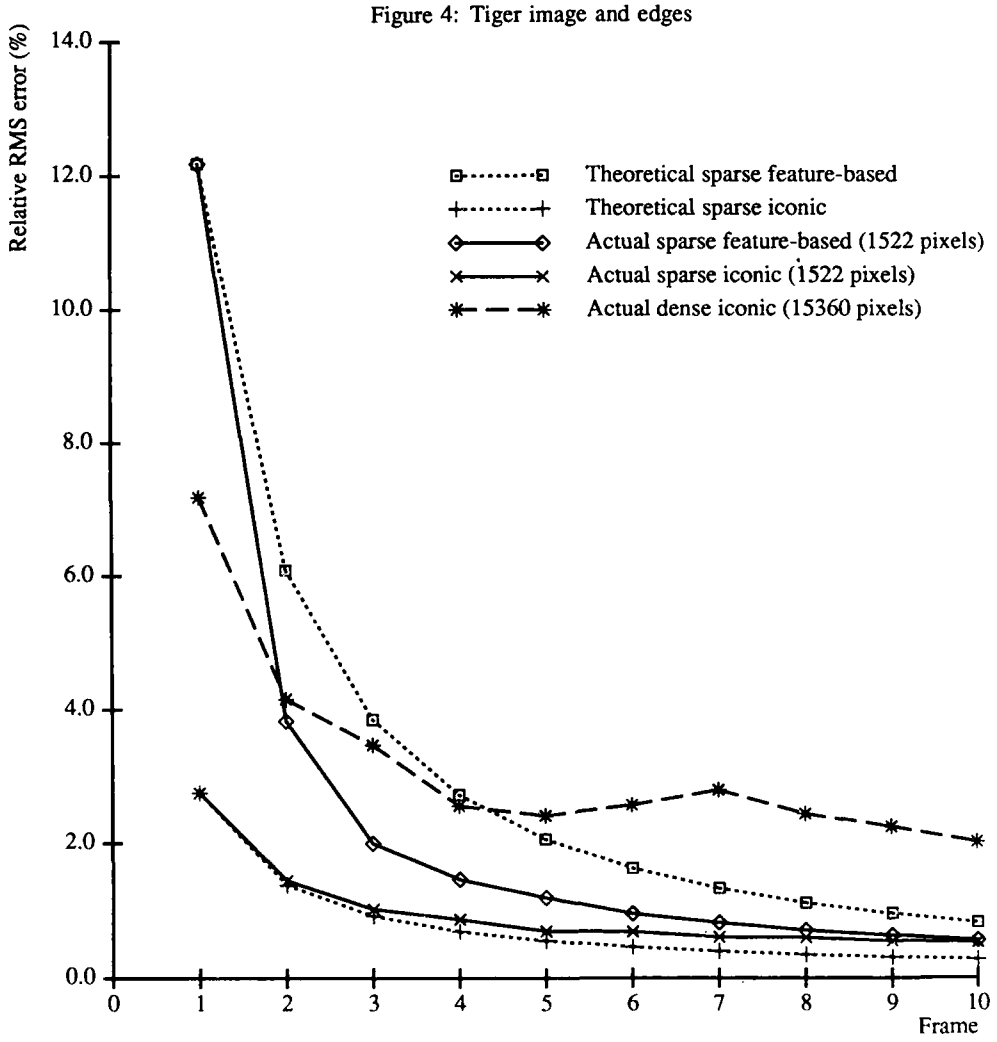



Figure 5: RMS error in depth estimate

an error level of approximately 0.5% percent after processing eleven images. Since the poster was 20 inches from the camera, this equates to a depth error of 0.1 inches. Note that the overall baseline between the first and the eleventh image was only 0.44 inches.

To compare the theoretical convergence rates derived earlier to the experimental rates, the theoretical curves were scaled to coincide with the experimental error after processing the first two frames. These scaled curves are also shown in Figure 5. For the iconic method, the theoretical rate plotted is the quadratic convergence predicted by the correlated flow measurement model. The agreement between theory and practice is quite good for the first three frames. Thereafter, the experimental RMS error decreases more slowly; this is probably due to the effects of unmodeled sources of noise. For the feature-based method, the experimental error initially decreases faster than predicted because the implementation required new edge matches to be consistent with the prior depth estimate. When this requirement was dropped, the results agreed very closely with the expected convergence rate.

Note that the comparison between theoretical and experimental results also allows us to estimate the precision of the sub-pixel edge extractor. The variance of a disparity estimate is twice the variance of the edge positions. Since the frame-to-frame displacement in this image sequence was one pixel and the relative RMS error was 12% for the first disparity estimate, the RMS error in edge localization was $0.12/\sqrt{2} \approx 0.09$ pixels.

Finally, Figure 5 also compares the RMS error for the sparse and dense depth estimates from the iconic method. The dense flow field is considerably noisier than the flow estimates that coincide with edges, though still just over two percent error by the end of eleven frames. Thus, the iconic method does provide valuable depth information at pixels not containing sharp edges.

### 5.3 Qualitative experiments: real scenes

We have tested the iconic and feature-based algorithms on complicated, realistic scenes obtained from the Calibrated Imaging Laboratory. Two sequences of ten images were taken with camera motion of 0.05 inches (1.27mm) between frames; one sequence moved the camera vertically, the other horizontally. The overall range of motion was therefore 0.5 inches (1.27 cm); this compares with distances to objects in the scene of 20 to 40 inches (51 to 102 cm).

Figures 6a-d shows one of the images, the edges extracted from it with an oriented Canny operator [Canny86], and depth maps produced by applying the iconic algorithm to the horizontal and vertical image sequences, respectively. Lighter areas in the depth maps are nearer. The main structure of the scene is recovered quite well in both cases, though the results with the horizontal sequence are considerably more noisy. This is most likely due to scanline jitter, as mentioned earlier. Edges oriented parallel to the direction of flow cause some scene structure to be observable in one sequence but not the other. This is most noticeable near the center of the scene, where a thin vertical object appears in Figure 6c but not in Figure 6d. This object corresponds to an antenna on the top of a foreground building (Figure 6a). In general, motion in orthogonal directions will yield more information than motion in any single direction.

Figure 7 shows intensity-coded depth maps and 3-D perspective reconstructions obtained with both the iconic and feature-based methods. These results were produced by combining disparity estimates from both horizontal and vertical camera motion. The depth map for the feature-based approach was produced from the sparse depth estimates by regularization. It is difficult to make quantitative statements about the performance of either method from this data, but qualitatively it is clear that both recover the structure of the scene quite well.

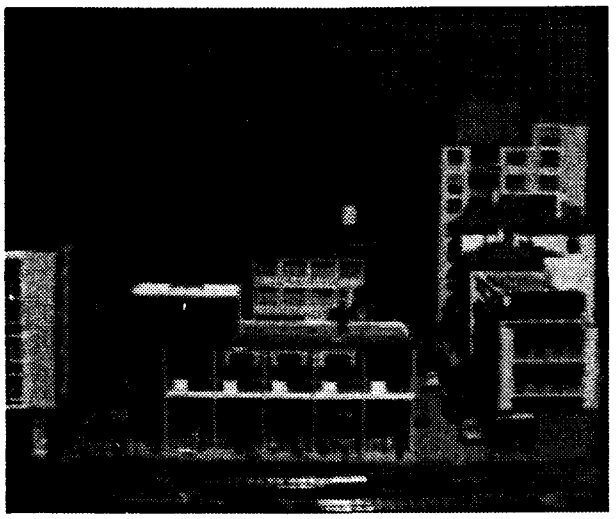
The iconic algorithm was also used to extract occluding boundaries from the depth map of Figure 6c (iconic method with vertical camera motion). We first computed an intrinsic "grazing angle" image [Matthies87d] giving the angle between the view vector through each pixel and the normal vector of the local 3-D surface. Edge detection and thresholding were applied to this image to find pixels where the view vector and the surface normal were nearly perpendicular. The resulting boundaries are shown along with the depth map in Figure 8. The method found most of the prominent building outlines and the outline of the bridge in the upper left.

## 6 Conclusions

This paper has presented a new algorithm for extracting depth from known motion. The algorithm processes an image sequence taken with small inter-frame displacements and produces an on-line estimate of depth that is refined over time. The algorithm produces a dense, iconic depth map and is suitable for implementation on parallel architectures.

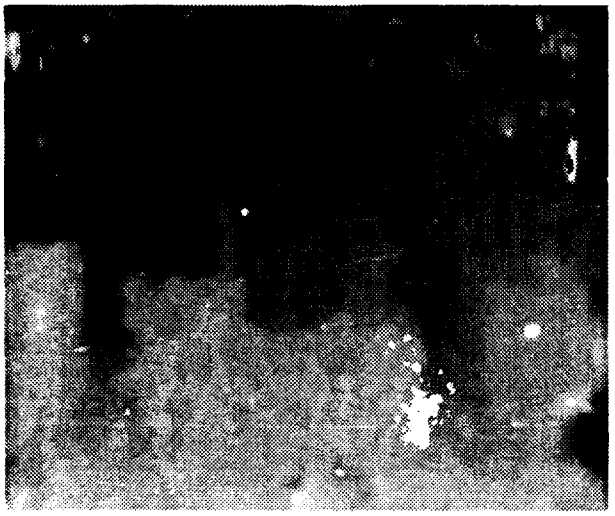
The on-line depth estimator is based on Kalman filtering. A correlation-based flow algorithm measures both the local displacement at each pixel and the confidence (or variance) of the displacement. These two "measurement images" are integrated with predicted depth and variance maps using a weighted least squares technique derived from the Kalman filter. Regularization-based smoothing is used to reduce the noise in the flow estimates and to fill in areas of unknown disparity. The current maps are extrapolated to the next frame by image warping, using the knowledge of the camera motion, and are resampled to keep the maps iconic.

The algorithm has been implemented, evaluated mathematically and experimentally, and compared with a feature-based algorithm that uses Kalman filtering to estimate the depth of edges. The mathematical analysis shows that the iconic approach will have a slower convergence rate because it only keeps one element of state per pixel (the disparity), while the feature-based approach keeps both the disparity and the sub-pixel position of the feature. However, an optimal implementation of the iconic method (which takes into account temporal correlations in the measurements) has the potential to equal the convergence rate and accuracy of the symbolic method. Experiments with images of a flat poster have confirmed this analysis and given quantitative measures of the performance



(a)

(b)

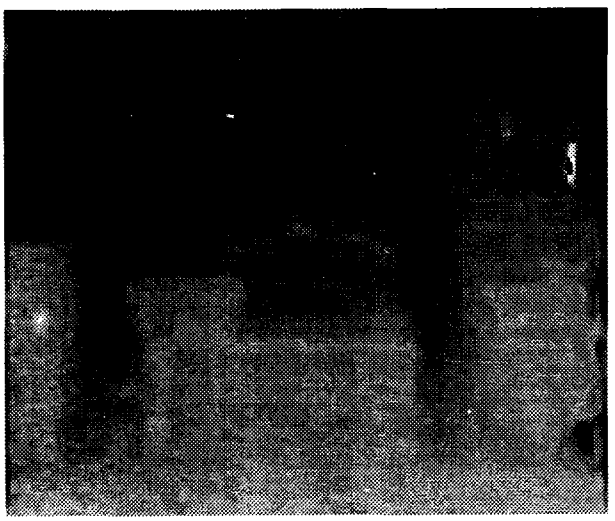


(c)

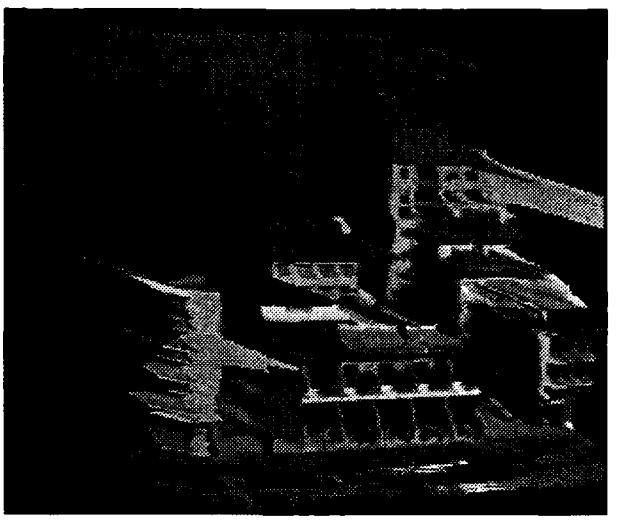
(d)

Figure 6: CIL depth maps

(a) first frame (b) edges (c) horizontal motion depth map (d) vertical motion depth map



(a)



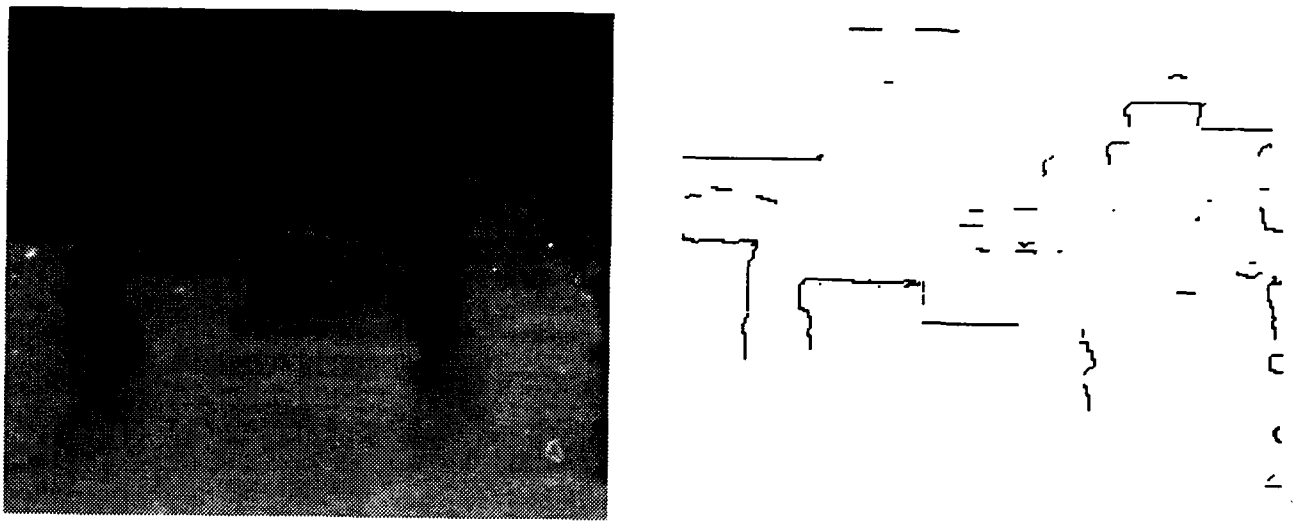
(b)

(c)

(d)

Figure 7: CIL orthogonal motion results

(a) iconic method depth map (b) perspective view (c) feature-based method depth map (d) perspective view



(a) (b)

Figure 8: Occluding boundaries
(a) vertical motion depth map (b) occluding boundaries

of both algorithms. Finally, experiments with images of a realistic outdoor scene model have shown that the new algorithm performs well on images with large variations in depth and that occluding boundaries can be extracted from the resulting depth maps.

### Extensions

The algorithms described in this paper can be extended in several ways. The most straightforward extension is to the case of non-lateral motion. As sketched in Section 3, this can be accomplished by designing a correlation-based flow estimator that produces two-dimensional flow vectors and an associated covariance matrix estimate [Anandan85]. This approach can also be used when the camera motion is uncertain or when the camera motion is variable (e.g. for widening baseline stereo [Xu85]). The alternative of searching only along epipolar lines during the correlation phase may be easier to implement, but is less general.

More research is required into the behavior of the correlation based flow and confidence estimator. In particular, we have observed that our current estimator produces biased estimates in the vicinity of intensity step edges. The correlation between spatially adjacent flow estimates, which is currently ignored, should be integrated into the Kalman filter framework. More sophisticated representations for the intensity and depth fields are also being investigated [Szeliski87b].

Finally, the incremental depth from motion algorithms we have developed can be used to initiate stereo fusion. Work is currently in progress investigating the integration of depth-from-motion and stereo [Matthies87a]. We believe that the framework presented in this paper will prove to be useful for integrating information from multiple visual sources and for tracking such information in a dynamic environment.

## References


[Anandan85] P. Anandan. Computing dense displacement fields with confidence measures in scenes containing occlusion. In *IUS Workshop*, pages 236–246, DARPA, December 1985.

[Ayache87] N. Ayache and O. D. Faugeras. Maintaining representations of the environment of a mobile robot. In *International Symposium of Robotics Research 4*, MIT Press, 1987.

[Baker87] H. H. Baker. Multiple-image computer vision. In *Proceedings of the 41st Photogrammetric Week*, pages 7–19, Stuttgart Institute for Photogrammetry, Stuttgart, West Germany, September 1987.

[Bolles87] R. C. Bolles, H. H. Baker, and D. H. Marimont. Epipolar-plane image analysis: an approach to determining structure from motion. *International Journal of Computer Vision*, 1:7–55, 1987.

[Broida86] T. J. Broida and R. Chellappa. Kinematics and structure of a rigid object from a sequence of noisy images. In *Proc. Workshop on Motion: Representation and Analysis*, pages 95–100, IEEE, May 1986.

[Canny86] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, November 1986.

[Faugeras86] O. D. Faugeras, N. Ayache, B. Faverjon, and F. Lustman. Building visual maps by combining noisy

stereo measurements. In *IEEE International Conference on Robotics and Automation*, pages 1433–1438, IEEE, San Francisco, California, April 1986.

[Gelb74] Technical Staff, The Analytic Sciences Corporation. *Applied Optimal Estimation*. MIT Press, Cambridge, MA, 1974.

[Gennery80] D.B. Gennery. *Modelling the environment of an exploring vehicle by means of stereo vision*. PhD thesis, Stanford University, June 1980.

[Hallam83] J. Hallam. Resolving observer motion by object tracking. In *International Joint Conference on Artificial Intelligence*, 1983.

[Heeger87] D. J. Heeger. Optical flow from spatiotemporal filters. In *First International Conference on Computer Vision*, pages 181–190, IEEE Computer Society Press, June 1987.

[Horn81] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[Matthies87a] L. H. Matthies. *Motion and Depth from Stereo Image Sequences*. PhD thesis, Carnegie Mellon University, (in preparation) 1987.

[Matthies87b] L. H. Matthies and T. Kanade. The cycle of uncertainty and constraint in robot perception. In *Proc. International Symposium on Robotics Research*, MIT Press, August 1987.

[Matthies87c] L. H. Matthies and S. A. Shafer. Error modeling in stereo navigation. *IEEE Journal of Robotics and Automation*, 239–248, June 1987.

[Matthies87d] L. H. Matthies, R. Szeliski, and T. Kanade. *Kalman Filter-based Algorithms for Estimating Depth from Image Sequences*. Technical Report CMU-CS-87-185, Computer Science Department, Carnegie Mellon University, December 1987.

[Maybeck79] P. S. Maybeck. *Stochastic Models, Estimation, and Control*. Volume 1, Academic Press, New York, NY, 1979.

[Maybeck82] P. S. Maybeck. *Stochastic Models, Estimation, and Control*. Volume 2, Academic Press, New York, NY, 1982.

[Mikhail76] E. M. Mikhail. *Observations and Least Squares*. University Press of America, Lanham, MD, 1976.

[Nagel86] H.-H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(5):565–593, September 1986.

[Nalwa86] V. Nalwa. On detecting edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):699–714, Nov. 1986.

[Rives86] P. Rives, E. Breuil, and B. Espiau. Recursive estimation of 3d features using optical flow and camera motion. In *Proceedings Conference on Intelligent Autonomous Systems*, pages 522–532, Elsevier Science Publishers, December 1986. (also appeared in Proc. 1987 IEEE Int'l Conf. on Robotics and Automation).

[Szeliski87a] R. Szeliski. Regularization uses fractal priors. In *Proceedings AAAI-87*, pages 749–754, Morgan Kaufmann Publishers, Seattle, Washington, July 1987.

[Szeliski87b] R. Szeliski. *Uncertainty in Low Level Representations*. PhD thesis, Carnegie Mellon University, (in preparation) 1987.

[Terzopoulos86a] D. Terzopoulos. Image analysis using multigrid relaxation methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(2):129–139, March 1986.

[Terzopoulos86b] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(4):413–424, July 1986.

[Xu85] G. Xu, S. Tsuji, and A. Minoru. Coarse-to-fine control strategy for matching motion stereo pairs. In *Proceedings of IJCAI*, pages 892–894, 1985.

# A Optic flow computation

In this appendix, we will analyze the variance of a simple correlation based flow estimator, the sum of squared differences (SSD) estimator [Anandan85]. An alternate approach to this analysis is taken in [Gennery80].

The SSD estimator selects at each pixel the disparity which minimizes the SSD measure

$$e(\tilde{d};x) = \int w(\lambda)[f_1(x+\tilde{d}+\lambda) - f_0(x+\lambda)]^2\, d\lambda,$$

where $f_0(x)$ and $f_1(x)$ are the two successive image frames and $w(x)$ is a symmetric, non-negative weighting function. To analyze its performance, we will assume that the two image frames are generated from an underlying true intensity image, $f(x)$, which is shifted by a true disparity $d$ in the second frame, and that uncorrelated (white) Gaussian noise with variance $\sigma_n^2$ is added to both frames:

$$\begin{aligned} f_0(x) &= f(x) + n_0(x), \\ f_1(x+d) &= f(x) + n_1(x). \end{aligned}$$

Using this model, we can rewrite the error measure as[3]

$$\begin{aligned} e(\tilde{d};x) = \int w(\lambda)[&f(x+\tilde{d}-d+\lambda) - f(x+\lambda) \\ &+n_1(x+\lambda) - n_0(x+\lambda)]^2\, d\lambda. \end{aligned}$$

If $\tilde{d} \simeq d$, we can use a Taylor series expansion to obtain

$$\begin{aligned} e(\tilde{d};x) = \int & w(\lambda)[f'(x+\lambda)](\tilde{d}-d)^2 \\ &+2w(\lambda)f'(x+\lambda) \quad [n_1(x+\lambda) - n_0(x+\lambda)](\tilde{d}-d) \\ &+w(\lambda)[n_1(x+\lambda) - n_0(x+\lambda)]^2\, d\lambda \\ &= a(x)(\tilde{d}-d)^2 + 2[b_1(x) - b_0(x)](\tilde{d}-d) + c(x), \quad (17) \end{aligned}$$

[3] This equation is actually incorrect, since it should contain $n_1(x+\tilde{d}\,d+)$ instead of $n_1(x+)$. The effect of including the correct term is to add small random terms involving integrals of $w()$, $w\,()$, $f\,(x+)$, $f\ (x+)$ and $n_1(x)$ to the quadratic coefficient $a(x)$, $b_1(x)$ and $c(x)$ that are derived below. This omission has been made to simplify the presentation.

where

$$a(x) = \int w(\lambda)[f'(x+\lambda)]^2\, d\lambda,$$

$$b_i(x) = \int w(\lambda) f'(x+\lambda) n_i(x+\lambda)\, d\lambda,$$

$$c(x) = \int w(\lambda)[n_1(x+\lambda) - n_0(x+\lambda)]^2\, d\lambda.$$

The four coefficients $a(x)$, $b_0(x)$, $b_1(x)$ and $c(x)$ define the shape of the error surface $e(\tilde{d}; x)$. The first coefficient, $a(x)$, is related to the average "roughness" or "slope" of the intensity surface; as shown below, this determines the confidence given to the disparity estimate. The second and third coefficients, $b_0(x)$ and $b_1(x)$, are independent zero mean Gaussian random variables that determine the error in the flow estimator. The fourth coefficient, $c(x)$, is a chi-squared distributed random variable with mean $(2\sigma_n^2 \int w(\lambda)\, d\lambda)$ that defines the computed error at $\tilde{d} = d$.

To estimate the disparity at point $x$ given the error surface $e(\tilde{d}; x)$, we find the $\hat{d}$ such that

$$e(\hat{d}; x) = \min_{\tilde{d}} e(\tilde{d}; x).$$

From the quadratic[4] equation in 17, we can compute $\hat{d}(x)$ as

$$\hat{d}(x) = d + \frac{b_0(x) - b_1(x)}{a(x)}.$$

To calculate the variance in this estimate, we must first calculate the variance in $b_i(x)$,

$$\mathrm{Var}(b_i(x)) = \sigma_n^2 \int w^2(\lambda)[f'(x+\lambda)]^2\, d\lambda.$$

If we set $w(x) = 1$ on some finite interval and zero elsewhere, this variance reduces to $\sigma_n^2 a(x)$ and we obtain

$$\mathrm{Var}(\hat{d}) = \frac{2\sigma_n^2}{a(x)}.$$

In addition to calculating the disparity estimate variance, we can compute its covariance with other estimates either in the same frame or in a subsequent frame. As described in Section 5.1, knowing the correlation between adjacent or successive measurements is important in obtaining good overall uncertainty estimates.

To determine the correlation between two adjacent disparity estimates, $\hat{d}(x)$ and $\hat{d}(x + \Delta x)$, we must first determine the correlation between $b_i(x)$ and $b_i(x + \Delta x)$,

$$\begin{aligned}\langle b_i(x) b_i(x+\Delta x)\rangle = & \\ & \int\int w(\lambda) w(\eta) f'(x+\lambda) f'(x+\Delta x+\eta) \\ & \langle n_i(x+\lambda) n_i(x+\Delta x+\eta)\rangle\, d\lambda\, d\eta \\ = & \int\int w(\lambda) w(\eta) f'(x+\lambda) f'(x+\Delta x+\eta) \\ & \delta(\lambda - \Delta x - \eta)\sigma_n^2\, d\lambda\, d\eta \\ = & \sigma_n^2 \int w(\lambda) w(\lambda - \Delta x)[f'(x+\lambda)]^2\, d\lambda.\end{aligned}$$

For a slowly varying gradient $f'(x)$, this correlation is proportional to the autocorrelation of the weighting function,

$$R_w(\Delta x) = \int w(\lambda) w(\lambda + \Delta x)\, d\lambda.$$

For the simple case of $w(x) = 1$ on $[-s, s]$, we obtain

$$R_{\hat{d}}(x, x+\Delta x) = \frac{2\sigma_n^2}{a(x)}\left(1 - \frac{|x|}{2s}\right) \quad \text{for} \quad |x| \le 2s.$$

The correlation between two successive measurements in time is easier to compute. Since

$$f_2(x + 2d) = f(x) + n_2(x),$$

we can show that the flow estimate obtained from the second pair of frames is

$$\hat{d}_2(x) = d + \frac{b_1(x) - b_2(x)}{a(x)}.$$

The covariance between $\hat{d}_1(x)$ and $\hat{d}_2(x)$ is

$$\mathrm{Cov}(\hat{d}_1(x), \hat{d}_2(x)) = \langle (\hat{d}_1(x) - d)(\hat{d}_2(x) - d)\rangle = -\frac{\sigma_n^2}{a(x)}$$

and the covariance matrix of the sequence of measurements $\hat{d}_t$ is

$$P_m = \frac{\sigma^2}{a}\begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & & & & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix}.$$

This structure is used in Section 5.1 to estimate the theoretical accuracy and convergence rate of the iconic depth from motion algorithm.

[4] The true equation (when higher order Taylor series terms are included) is a polynomial series in $(\tilde{d}\ d)$ with random coefficients of decreasing variance. This explains the "rough" nature of the $e(\tilde{d}; x)$ observed in practice.

# MULTIMODAL RECONSTRUCTION AND SEGMENTATION WITH MARKOV RANDOM FIELDS AND HCF OPTIMIZATION

P.B. Chou and C.M. Brown

Computer Science Department
University of Rochester
Rochester, New York 14627

**Abstract**

Segmentation of a visual input into regions corresponding to three-dimensional objects is an important step in many image analysis algorithms. Reconstruction of physical properties (such as depth or surface orientation) is similarly important. Fusing multiple sources of visual data seems a practical way to improve robustness, as does guiding interpretation with prior domain knowledge. Coupled Markov random fields provide a unifying mechanism to accomplish all these goals simultaneously. The Highest Confidence First algorithm is an optimization heuristic that yields good results in finding high-probability reconstruction and discontinuity values. Its use in this context involves making precise the relation between intensity and depth discontinuties. In synthetic data, the contribution of intensity information to depth reconstruction and segmentation is significant.

## 1. Introduction

The reconstruction of three-dimensional scene parameters (intrinsic images) from visual information is often accomplished using a smoothness assumption to regularize the computation. Smoothing is not wanted across object boundaries, and reliable reconstruction can not be achieved without the detection of the discontinuities [Stuth, Ballard, and Brown 1983]. Thus the cooperation of reconstruction and discontinuity detection has been of interest for some time: The challenge is to develop a unified treatment for reconstruction and segmentation. Much recent work on cooperative reconstruction has two important characteristics: It uses global measures and is thus robust against local noise, and it explicitly incorporates prior knowledge to solve the ill-posed reconstruction problem [Blake and Zisserman 1987] [Marroquin, Mitter, and Poggio 1985] [Torre and Poggio 1986] [Geman and Geman 1984] [Chou and Brown 1987].

Integrating disparate sources of information has been recognized as one of the keys to the success of general purpose vision systems. It has been proposed that integration occurs naturally during the reconstruction of the visible surfaces [Marr 1982], and is best performed at the locations of discontinuities [Gamble and Poggio 1987]. For example, one problem may be to reconstruct a depth map and to detect its discontinuities simultaneously from a (sparse) set of corrupted depth observations, possibly with the aid of other sources of information. This paper reports our recent work in incorporating intensity discontinuity observations to reconstruct and segment such a depth map. We use Markov Random Fields (MRF's) to model the probabilistic relations between image events. Probabilistic models of depth measurements and intensity discontinuities enable a Bayesian approach to combine them consistently with *a priori* knowledge. We use the Highest Confidence First (HCF) method [Chou, Brown, and Raman 1987] to approximate the Maximum A Posteriori estimates. Preliminary experiments show that sparse (50% density) depth data and intensity can be reliably combined using HCF. The probabilistic approach is flexible -- it suggests a general framework for integrating various visual cues such as motion, texture, and albedo, and detection of features such as edges and corners with *a priori* knowledge to obtain robust multi-modal reconstruction and segmentation.

### 1.1. Related Research

Much work on visual reconstruction, although it may have common goals and computations, can be described from two different viewpoints, according to the underlying surface models. In both cases a form of energy minimization is performed to obtain the final state of the system, which results in assigning continuous-valued labels such as depth values, and discrete labels such as "discontinuity" or "no discontinuity" to elements in the system. A common characteristic of the approaches to visual reconstruction is the use of cooperative networks as the model of computation for energy minimization. In a cooperative network, each computation unit (site) performs simple operations. The connections between the sites usually reflect direct dependencies between them. Collectively, the outputs (labels) of the sites, at a stable system state, correspond to the solutions of the underlying problem.

The *mechanical viewpoint* models smooth surfaces with membranes and thin plates [Grimson 1984] [Terzopoulos 1986]; Terzopoulos employs a mechanical spring/surface model to integrate depth, orientation, and discontinuity constraints. The depth and orientation measurements act as springs constraining the configurations of plates and membranes; the spring constants represent the confidence associated with the corresponding measurements. The depth and orientation discontinuities, if detected prior to the integration, indicate the locations where the assumed properties are violated. Discontinuities can also be detected as locations of abnormally high strain in the modeled surface. The most successful computational approach in this formulation has been a coarse-to-fine resolution computation [Terzopoulos 1983].

The *probabilistic viewpoint* encodes *a priori* beliefs about the world in terms of probabilistic distributions - MRF/Gibbs distributions in particular [Geman and Geman 1984] [Marroquin, Mitter, and Poggio 1985]. Good overviews of this approach are given in [Marroquin 1985]. The details of MRFs and their relation to probabilities and energy models will not be rehearsed here. Qualitatively, they capture the locality of interaction that seems relevant for low-level vision. The Hammersley-Clifford theorem relates the configuration of the MRF to the well-known Gibbs energy distribution, and provides a computational method for calculating stable states of the field using only local information for each site. MRFs can incorporate *a priori* knowledge, and depending on the loss energy function used, they can maximize, given the input information, the A Posteriori probability (MAP estimation) or the A Posteriori Marginal probability (MPM estimation). Computing the minimal energy (maximum probability)

state of an MRF is a difficult problem in optimization, to which many techniques have been applied, including simple deterministic gradient-space approaches and sophisticated simulated annealing techniques. The Highest Confidence First technique [Chou and Brown 1987] is a deterministic gradient descent method in an augmented space of site labels. There is an extra "uncommitted" label, and a scheduling algorithm that labels confident sites early. The results are good, and the time needed is predictable and competitively small.

In fusing depth and intensity information, Gamble and Poggio [Gamble and Poggio 1987] use the intensity edges detected with the Canny operator to constrain the locations of the depth discontinuities while reconstructing a depth map. Their rule is that no depth discontinuity is allowed without a corresponding intensity discontinuity. The results of combining the two information modalities are encouraging and better than either modality operating alone, but the uncompromising relation between depth and intensity discontinuities means that depth discontinuities within regions of little intensity variation will be lost even if the depth information is good. To use the HCF algorithm to fuse depth and intensity, three issues arise: how to specify the energy functional for the continuous depth label, how to specify the "confidence" or "stability" of a site for the HCF algorithm, and how to assign a general *a priori* relation between depth and intensity information. For the last issue, Chou and Brown [Chou and Brown 1987] treat visual integration in the context of the labeling problem. In this paper we organize interesting labels as a hierarchically structured tree, where bodies of external evidence, in terms of likelihoods of the labels, are combined following Bayes' rule and a couple of conditional independence assumptions. The combined evidence is fused with spatial prior knowledge at an appropriate level of the tree resulting in the *a posteriori* distribution for the labeling problem.

In the remainder of this paper, we give technical details of the formulation of the fusion problem and present some experimental results and directions of future work

## 2. Coupled MRF's and Observation Models

Represent a pixel image $S = \{s_1, s_2, \ldots, s_N\}$ as a set of grid-structured sites, and the discontinuity image, $D$, as the set of sites placed midway between each vertical and horizontal pair of pixel sites. Let $F = \{f_s, s \in S\}$ be the set of random variables indexed by S, with $f_s \in R$ representing the depth value at location $s$, and $L = \{l_d, d \in D\}$ be the set of random variables indexed by $D$, with $l_d \in \{0,1\}$ representing the absence or presence of a discontinuity at site $d$. $F$ and $L$ correspond to the depth (intensity) process and the line process respectively of the coupled Markov Random Fields introduced by Geman and Geman [Geman and Geman 1984] and used in [Marroquin, Mitter, and Poggio 1985], [Gamble and Poggio 1987]., and this paper. A configuration of $(F,L)$ corresponds to an admissible solution to our problem.

### 2.1. Early Visual Observations

We model early visual computations as the computations performed by a set of independent modules. These modules produce estimates of surface parameters and discontinuities either through active sensing, such as radar and laser, or with passive visual data, such as image irradiance and stereo disparities. A module, when making an opinion (hereafter, observation) about a site $t$, examines its input at some small spatial region dependent on $t$. It is reasonable to assume that the characteristics of these modules can be modeled with statistical knowledge of domain noise and sensor error. Hence their observations of image parameters can be represented in terms of likelihoods or likelihood ratios [Sher 1987] [Chou and Brown 1987], reflecting how likely the observation is, given a state of the scene.

*Early Depth Measurements*: The measurements are considered sparse and independently measured. Denote by $\hat{S}$ the set of sites in $S$ at which depth measurements are available and $g$ the set of these measurements. We assume

$$P(F = g \mid F = f) = \prod_{s \in \hat{S}} P_s(g_s \mid f_s) \tag{2.1}$$

where $g_s$ denotes the measurement at pixel site $s$. Often the noise can be adequately modeled by unbiased Gaussian distributions. That is

$$P_s(g_s \mid f_s) = \frac{1}{z} e^{\frac{-(f_s - g_s)^2}{2\sigma_s^2}} \tag{2.2}$$

*Discontinuity Observations Based on Intensity*: Instead of treating intensity edges as constraints on the locations of depth discontinuities, we consider them as partial evidence supporting or refuting the hypotheses about depth discontinuities. The motivation is simple. The intensity images are the results of many confounding factors - lighting, surface geometry, surface reflectance, and camera characteristics. Intensity discontinuities may reflect sudden changes of depth values, but depth discontinuities do not necessarily imply large intensity variations. Figure 1 shows the conceptual hierarchy that consists of the interesting events involved here. At the first level, only EDGE or NON-EDGE is of concern. Node EDGE represents the event that the site of interest corresponds to some sort of discontinuity in the world; NON-EDGE represents the event that the site is within a homogeneous region. At the next level, whether a particular discontinuity is due to depth discontinuation becomes interesting. Intensity observations provide information about the events in the first level, but say nothing about the events in the second level, which are important to the depth segmentation problem. Our approach is to incorporate prior experience and knowledge, represented in terms of conditional probabilities, to infer the amount of support, represented as likelihood ratios, to the events of depth discontinuities provided by the intensity observations. Given the conditional probabilities corresponding to the father-son links in the figure, the likelihood ratio of DEPTH-EDGE can be calculated [Chou and Brown 1987]. In the rest of the paper, $\lambda_d$ denotes the likelihood ratio of site $d$ given the intensity observation $O_d$, where $d \in D$.

$$\lambda_d(l_d) = \frac{P(O_d \mid l_d)}{P(O_d \mid \neg l_d)} . \tag{2.3}$$

Again, we consider the spatially distinct intensity observations are conditionally independent:

$$P(O \mid l) = \prod_{d \in D} P_d(O_d \mid l_d) , \tag{2.4}$$

where $O$ denotes the collection of intensity observations.

*Conditional Independence between Intensity and Depth Observations*: We assume that the depth and intensity observations are only related through the geometry of the surfaces in view. They are conditionally independent in the following sense:

$$P(g, O \mid f, l) = P(g \mid f, l)\, P(O \mid f, l) . \tag{2.5a}$$

We further assume that the knowledge of depth discontinuities contributes no information to make one prefer the observation of $f$ over others once the true depth values are known:

$$P(g \mid f, l) = P(g \mid f) , \tag{2.5b}$$

and that the knowledge of surface depth does not make $O$ more or less likely once the depth discontinuities are known:

$$P(O \mid f, l) = P(O \mid l) . \tag{2.5c}$$

The scene depth, in many circumstances, affects the observed

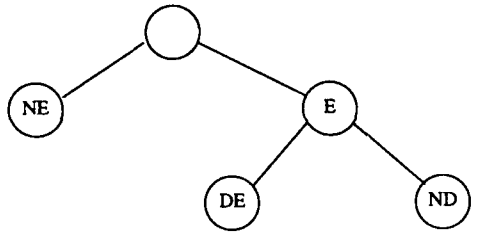



Figure 1: A label hierarchy. Label E denotes an edge element (discontinuity), NE non-edge, DE depth edge, and ND edge but not depth edge.

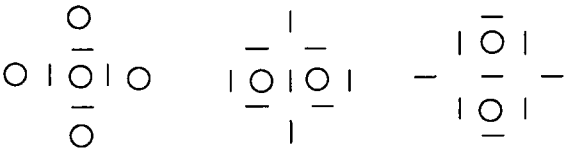


Figure 2: Neighborhood system used. Circles represent pixel sites; lines represent discontinuity sites. Surrounding sites are the neighbors of the central site.

intensity values. The assumption (2.5c) is reasonable, however, since it is the indirect observations of intensity discontinuities but not the magnitude of the intensity that are actually used in this work. Thus in this work we discard intensities after computing likelihoods of discontinuities. An interesting research problem would be to use the intensity information (perhaps through the irradiance-orientation constraint [Horn 1975]) more directly.

Summarizing (2.1) - (2.5), we assume

$$P(\boldsymbol{g},\boldsymbol{O}\,|\,f,l) = \prod_{s\in S} P_s(g_s\,|\,f_s) \prod_{d\in D} P_d(O_d\,|\,l_d)\ . \tag{2.6}$$

### 2.2. Markov Random Fields and Energy Measures

In short, a Markov Random Field is a set of random variables such that the value of a variable depends, probabilistically, on the values of its neighboring variables. The neighbor relation, an important part of its definition, encodes the only direct interdependence among the variables. The utility of the MRF concept is that the dependencies among a large number of variables can be adequately modeled with neighborhoods that are small enough for practical purposes.

Spatial adjacency is a natural neighboring relationship. Within each $F$ and $L$, spatially adjacent variables tend to have similar values. That is, surfaces and boundaries tend to be continuous and smooth. MRF's corresponding to $F$ and $L$ can be separately defined to model these properties. Chou et al [Chou, Brown, and Raman 1987] have demonstrated some promising edge detection results using an MRF for the line process alone. The depth and line processes, however, are not independent of each other. The presence of a line at an edge site breaks the connection between the two variables at the adjacent pixel sites; a small change in the values of two adjacent depth variables suggests the absence of a discontinuity in between. This interdependence is the basis for the concept of coupled MRF's - an unified treatment of reconstruction and segmentation. Figure 2 shows the neighborhood system $\Gamma$ of the MRF consisting of the edge and line processes used in our experiments. In addition to the depth and line processes, the concept of coupled MRF's can also be applied to model many other interdependent processes corresponding to various intrinsic parameters [Poggio 1985].

$(F,L)$ is an MRF with respect to the neighborhood system $\Gamma$ if and only if, according to Hammersley-Clifford theorem, the joint probability distribution of the variables is a Gibbs distribution. That is,

$$P(f,l) = \frac{1}{Z}\, e^{-\frac{U(f,l)}{T}}\ , \tag{2.7a}$$

where Z is a normalization constant, $T$ the "temperature" of the field, and the energy functional

$$U(f,l) = \sum_{c\in C} V_c(f,l)\ . \tag{2.7b}$$

C denotes the set of cliques defined by $\Gamma$. $V_c$, the potential functional of clique $c$, measures the contribution to the total energy from the local interactions of the elements in $c$. The MRF-Gibbs equivalence not only relates the local conditional probabilities to the global joint probabilities, but also provides us a conceptually simpler way of specifying MRF's, thus the prior knowledge, by specifying potentials. For example, continuous surfaces can be modeled by setting the potential energy for the cliques consisting of two adjacent depth sites, say $i$ and $j$, and the line site in between them , say $ij$, proportional to $(1-l_{ij})(f_i-f_j)^2$ [Marroquin, Mitter, and Poggio 1985] [Gamble and Poggio 1987]. Other types of cliques that have non-zero potential functionals used in our experiments consist only of line sites. They are described in [Chou, Brown, and Raman 1987].

### 2.3. A Posteriori Energy

Bayes' rule combines the *a priori* knowledge and the early visual observations to derive the *a posteriori* belief.

$$P(f,l\,|\,\boldsymbol{g},\boldsymbol{O}) = \frac{P(f,l)P(\boldsymbol{g},\boldsymbol{O}\,|\,f,l)}{\sum_{f,l} P(f,l)P(\boldsymbol{g},\boldsymbol{O}\,|\,f,l)}\ . \tag{Bayes}$$

Note, from (2.1) and (2.4), that scaling all of the likelihoods for a fixed site by a constant does not change the posterior distribution of $(F,L)$. From (2.6) and (2.7), and assuming (2.2), the posterior distribution is a Gibbs distribution, with the *a posteriori* energy functional

$$U(f,l\,|\,\boldsymbol{g},\boldsymbol{O}) = \sum_{c\in C} V_c(f,l) + T\Big(\sum_{s\in S} \frac{(f_s-g_s)^2}{2\sigma_s^{\,2}} - \sum_{d\in D} \log\lambda_d(l_d)\Big) \tag{2.8}$$

## 3. Energy Minimization with HCF

One important aspect of the Highest Confidence First method is the use of an "uncommitted" state for each site. An uncommitted site is a node with only input links in a cooperative network in the sense that it does not contribute to the decisions of other sites while collecting information from its committed neighbors. The uncommitment reflects the lack of "confidence" to make a decision based on its current knowledge. Only when no other sites have higher confidence will an uncommitted site commit to its best choice. Commitments are not final: Any commitment can be altered later if strong opposition is posted by others.

Let $\zeta$ denote the uncommitted state, and $\bar{R} = R\cup\{\zeta\}$, $\bar{L} = \{\zeta,0,1\}$ denote the *augmented state spaces* for the depth and line processes respectively. Based on (2.8), define the *augmented local energy* measures with respect to an augmented configuration $(f,l)$ as:

$$E_s(f) = \sum_{c:\, s\in c} V'_c(f',l) + T\,\frac{(f_s-g_s)^2}{2\sigma_s^2} \quad \text{for } s\in\hat{S}\,, \tag{3.1a}$$

$$E_s(f) = \sum_{c:\, s\in c} V'_c(f',l) \quad \text{for } s\in S-\hat{S}\,, \tag{3.1b}$$

and

$$E_d(l) = \sum_{c:\, d\in c} V'_c(f,l') - T\log\lambda_d(l) \quad \text{for } d\in D\,, \tag{3.1c}$$

where $(f',l')$ agree with $(f,l)$ everywhere except $f'_s = f$ and $l'_d = l$, with $(f,l)\in(R,L)$. $V'_c = 0$ if there is an uncommitted site in $c$, otherwise it is equal to $V_c$. Thus the cliques containing uncommitted sites have no effect on the augmented energy measures. Since the only cliques involved in (3.1a) and (3.1b) are those consisting of two neighboring pixel sites and a line site in between, the terms $\sum_{c:\, s\in c} V'_c(f',l)$ can be written as $\sum_{r\in N_s} \beta(1-l_{rs})\,(f_r-f_s)^2$, where $N_s$ is the pixel neighborhood of $s$. The augmented local energy measures for pixel sites thus are quadratic; the shape of each quadratic depends on the constant parameter $\beta$, the number of active neighbors, and the variances of the noise in the early measurements. The temperature $T$ is set to 1 throughout our experiments, therefore $\beta$ decides the degree of smoothness relative to the magnitude of noise.

### 3.1. Stability Measures

The confidence of a site in a configuration $(f,l)$ is evaluated in terms of the following *tability* measures:

$$\begin{aligned} G_s(f,l) &= \Delta E_s(f_{\min}, f_{\min}+\alpha) \quad \text{if } f_s=\zeta\,, \\ &= \Delta E_s(f_{\min}, f_s) \qquad \text{otherwise}\,, \end{aligned} \tag{3.2a}$$

where $E_s(f_{\min}) = \min_{f\in R} E_s(f)$, and

$$\begin{aligned} G_d(f,l) &= \max_{l\in L,\, l\neq l_{\min}} \Delta E_d(l_{\min}, l) \quad \text{if } l_d=\zeta\,, \\ &= \Delta E_d(l_{\min}, l_d) \qquad \text{otherwise}\,, \end{aligned} \tag{3.2b}$$

where $E_d(l_{\min}) = \min_{l\in L} E_d(l)$. The term $\Delta E_r(k,j)$ is defined as $E_r(k) - E_r(j)$ with respect to $(f,l)$. It represents the change in local energy measure of $r$, thus the global energy (e.g. (2.8)), if $r$ should switch its state from $j$ to $k$. The stability of a site is nonnegative only when it is in its minimal energy state (i.e. $l_{\min}$ or $f_{\min}$) with respect to its current local energy measure. A large negative stability value signals high confidence in making a state change to the minimal energy state. The constant $\alpha$ determines the stability of uncommitted pixel sites: it gives the price paid in energy for remaining uncommitted. $\alpha$ has the semantics of offset along $R$ from state of minimal energy. Using it, the stability measure for uncommitted states has the semantics "how much energy could be lost by committing." Large $\alpha$ encourages quicker commitment.

### 3.2. Convergence Properties

The HCF network behaves as follows. Every site starts in the uncommitted state. At any instant, only the sites with the highest confidence in changing their states, i.e., the least stable ones with respect to the current configuration, are allowed to change their states. The identities of the sites, pixel or discontinuity, are ignored in the process of comparing the stability measures. Thus the reconstruction and segmentation processes proceed simultaneously. Eventually, the network settles at a configuration when no further reduction of the global energy measure can be made at each site; i.e., when all local stability measures are nonnegative.

The convergence property can be easily verified [Chou and Raman 1987]. It is possible, however, that the final configuration contains uncommitted depth sites, due to the sparseness of the depth data. In the initial stage of the computation, $E_s(f) = 0$ if $s\in S-\hat{S}$. This local energy measure, and thus the stability measure, will remain zero until one of the pixel neighbors becomes committed and the line process indicates there is no discontinuity in between. If a region of pixel sites, consisting only of members of $S-\hat{S}$, was surrounded with discontinuities before any of them has nonzero stability measure, all of them will stay uncommitted. In the extreme case, if there is no depth measurement at all, this network will not produce an estimate of the depth map. We think that as an advantageous feature since in such degenerate cases, there are infinite number of configurations that have the minimal energy measure. It is important for the low-level process to indicate the lack of information to higher-level processes so that attention can be directed to acquire more information. This feature can be turned off, if desired, by assigning *a priori* estimates (e.g. expected range of the scene) to those sites.

An interactive graphical simulator for several MRF optimization techniques was developed for our original work with line process [Chou and Raman 1987]. An enhanced version was built for experiments on coupled depth and intensity fields with HCF optimization. We implemented the HCF network on a serial machine using a binary heap to decide the visiting order of the sites. At any instant, the top element of the heap is the site with the smallest stability measure. A state change made by a site in general changes the stability measures of the sites in its neighborhood. The number of comparisons in maintaining the heap property for each change is limited by the height of the heap -- $\approx \log_2(3N)$ where $N$ is the number of sites. Ideally, the computation terminates when the top element has nonnegative stability since no more energy reduction would be possible afterwards. In practice, a small (negative) threshold is used to force termination without noticeable degradation of depth value (see below). Some threshold would have been necessary in any case because of limited precision in the calculations.

## 4. Results

The enhanced HCF algorithm, which reconstructs depth and finds depth discontinuities from a pair of depth and intensity images, is demonstrated on two synthetic scenes. Each scene consists of a range image and an irradiance image. Noise of a particular description is added independently to each image. The range image is sampled either at full resolution or randomly at reduced resolution (this section reports experiments with 100%, 80%, and 50% of the original full-resolution data points kept in the sparse data.)

The first sequence shows experiments with images created by utilities in the PADL-2 solid modeling system [Brown 1982] Figs. 3a and b show original full-resolution intensity and depth images, and Figs. 3c and d show the images with added noise. The intensity image has a range of pixel values in [0,255], and is perturbed by zero-mean, signal-independent gaussian-distributed noise $G(0, \sigma)$ with $\sigma=16$. Perturbed values less than 0 are set to 0, greater than 255 are set to 255. The range image has an unusual noise pattern. Part of the motivation is to test the effects of spatially nonuniform noise (the noise model affects the stability calculations of elements). Another motivation is to reflect a range imaging system whose values are more accurate near its optic axis, perhaps as an effect of reduced resolution (averaging) in the periphery. The noise distribution for the synthetic range image is radially symmetric around the center of the image, with standard deviation of the additive, signal-independent, zero-mean gaussian noise at a point increasing exponentially as the distance of the point from the center of the image. The exponential is scaled so the maximum noise has $\sigma=20$ (in the corners of the image.)

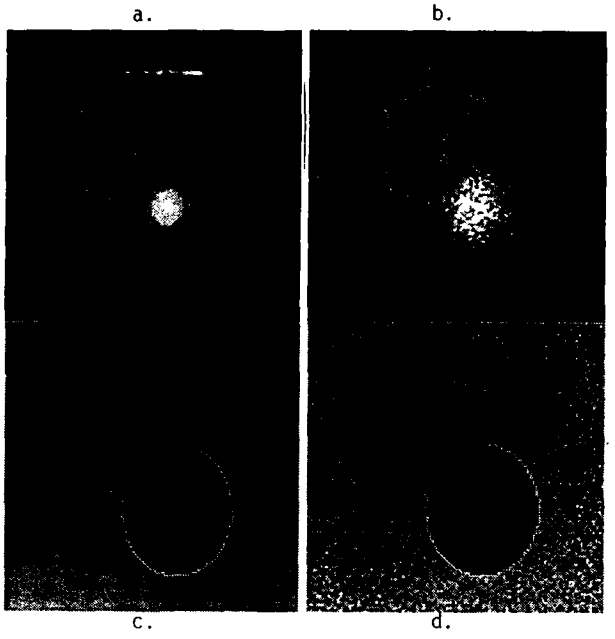



Fig. 3: a) Original intensity image. b) Original depth image. c) Intensity image with G(0,20) additive noise, clipped to range [0,255]. d) Depth image with spatially-varying Gaussian noise, maximum standard deviation of 20 (see text).

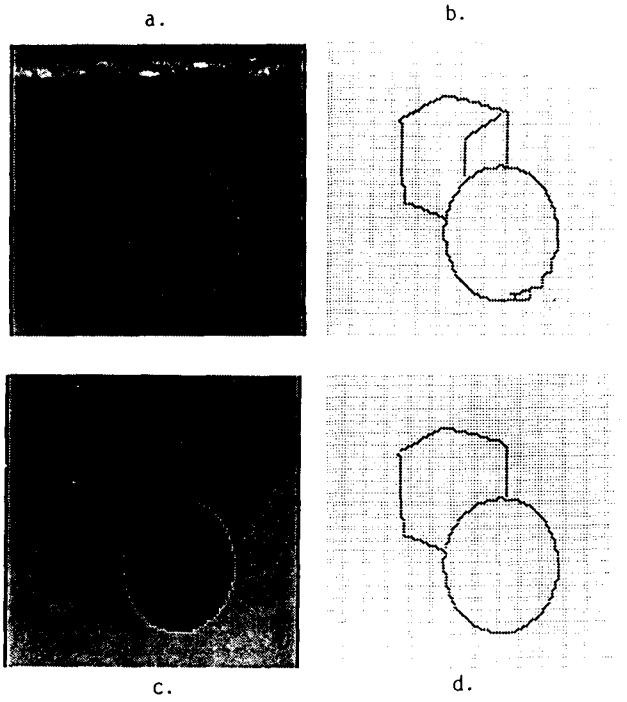



Fig. 4: a) Reconstructed depth with $\alpha = 10$ and $\beta = 0.01$. b) Depth edges with $\alpha$ and $\beta$ as in a). c) Reconstructed depth with $\alpha = 10$ and $\beta = 0.001$. d) Depth edges with $\alpha$ and $\beta$ as in c).

Figs. 4a and b show the reconstructed depth and the depth discontinuities found with a high setting of the parameter $\beta$, which increases sensitivity to small depth discontinuities. Here orientation changes in the cube and noise in the sphere boundary have given rise to a spurious segmentation. In Figs. 4c and d, a smaller $\beta$ avoids the problems.

Figs. 5 and 6 illustrate the effect of sparse depth data on the final depth reconstruction and segmentation, and also show the beneficial effects of incorporating intensity information. In these four experiments $\alpha$ and $\beta$ are held constant. Figs. 5a and b show the reconstruction and segmentation using just 80% of the depth information. It is interesting that the results are no worse; however Figs. 5c and d show the improvement gained by allowing the MRF access to the irradiance image as well. Fig. 6 is precisely analogous to Fig. 5, only the depth density is down to 50%.

Figs. 7a and b show the "depth" and "irradiance" parts of an artificial scene of the sort used by Sher [Sher 1987]. Such scenes have the advantage of having well-specified right and wrong locations for edges. In this case both depth and intensity images have spatially-independent, zero-mean, signal-independent, additive gaussian noise, and the perturbed image values are clipped to the range [0, 255]. For the depth image, $\sigma = 16$, and for the intensity image $\sigma = 20$. Fig. 7c shows the depth edges recovered using only the depth image (Fig. 7a), and Fig. 7d shows the depth edges recovered using depth (full resolution) and intensity. As expected, 7c only shows and edge structure related to the brightness differences of squares in Fig. 7a. Fig. 7d has clearly incorporated information from the intensity image Fig. 7b. The ideal desired is of course that all lines of the checkerboard are in evidence. It can be seen that the lines are missed when the evidence in both images is weak.

Fig. 8 shows the effects of sparse depth in the domain of Fig. 7. Fig. 8a gives a glimpse into the inner state of the MRF

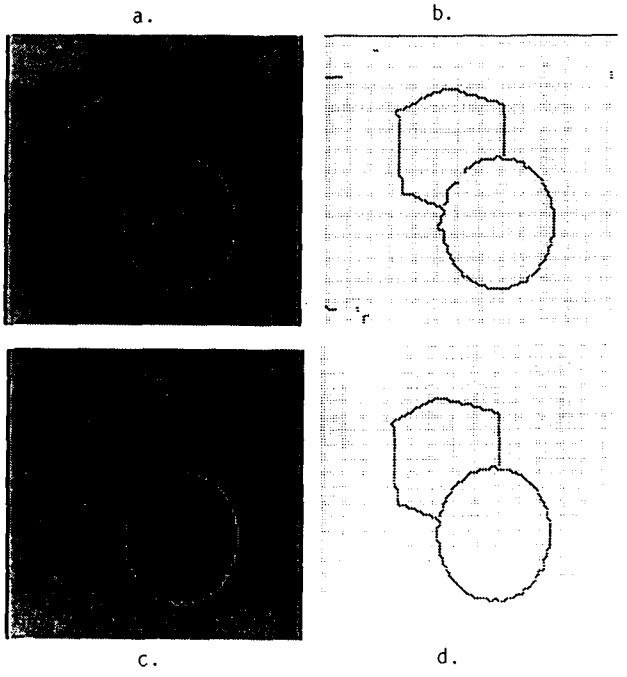



Fig. 5: a) Reconstructed depth with $\alpha = 50$ and $\beta = 0.001$, with 80% depth data randomly sampled and no intensity input. b) Depth edges with conditions of a). c) Reconstructed depth with $\alpha$ and $\beta$ and 80% depth sampling as in a), but using intensity input in MRF. d) Depth edges with conditions of c)

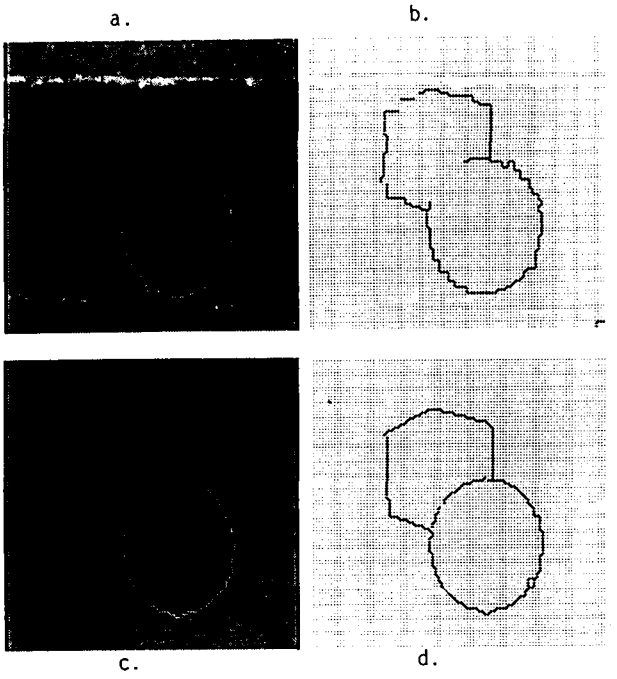



Fig. 6: a), b), c), d) as in Fig. 5 except that depth sampling density is 50%.

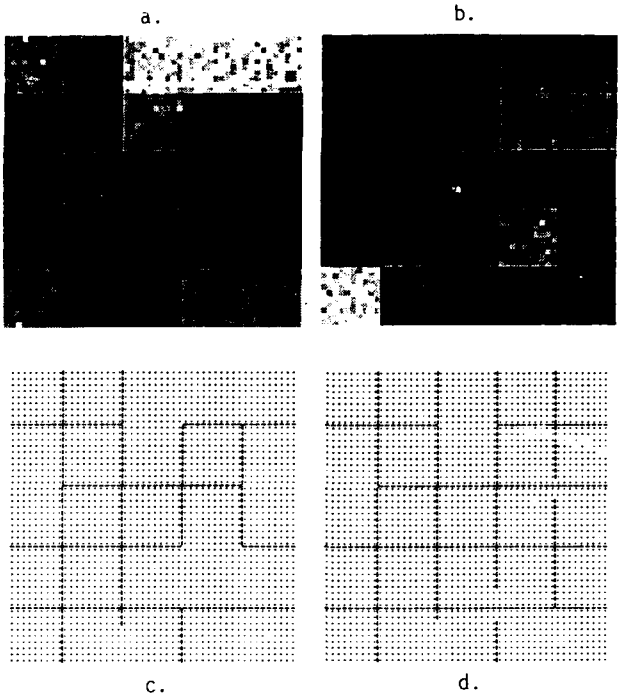



Fig. 7: a) Artificial "depth" image, with spatially independent G(0,16) additive noise, clipped to [0,255]. b) Artificial "intensity" image, with G(0,20) noise as in a).a c) Depth edges found with $\alpha = 50$, $\beta = 0.001$, no intensity input used in MRF. d) Depth edges found under conditions of c) but using intensity input.

algorithm. It shows the initial thresholded likelihoods (line elements) used by the intensity discontinuity detector, overlaid on the points where sparse (50%) depth information is available. Figs. 8b and c show the reconstructed depth and depth discontinuities, respectively. The areas of bad performance correlate with areas of weak or missing information.

## 5. Discussion and Future Work

### 5.1. Discussion

*The Role of Intensity Discontinuities*: Successful integration of multi-modal data requires knowledge about the characteristics of scene and the vision modules processing the data. Such knowledge affects the decisions that have to be made when different modalities provide conflicting information about particular events. Fig. 3 shows an example: The strong intensity gradients across the cube edges suggest depth discontinuities at the face intersections but the relatively small depth differences at these locations refute such suggestions. Also, the self-shadowed face merges with the background in the intensity image while the depth information indicates clear separation of the two regions. If reliable depth observations are available, e.g., a noise free depth map, it is bad practice to use the intensity observations for clues to depth discontinuities. On the other hand, when the depth observations are sparse and unreliable, the correlation of intensity information with depth information should be recognized and used. The probabilistic integration provided by HCF optimization is one coherent framework for such integration tasks. The HCF scheme, as a deterministic method, finds a local probability maximum. In so doing, its behavior is consistent with the natural evidence weighing described above. In

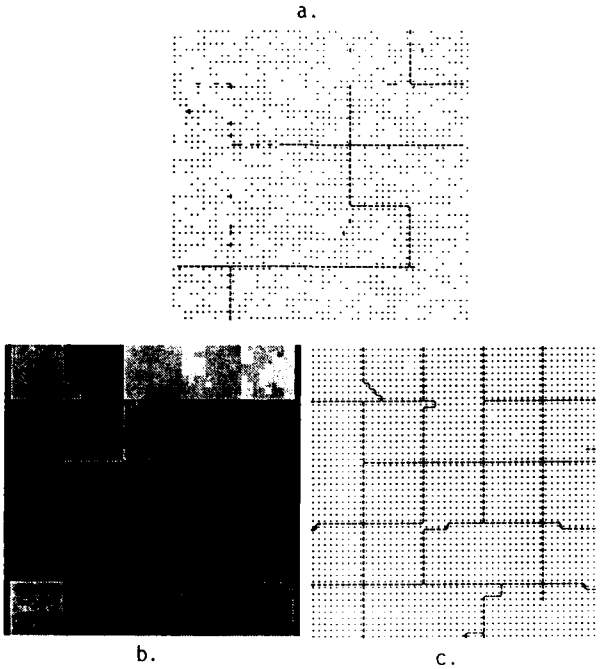



Fig. 8: a) Locations of 50% depth sampling overlaid with the initial thresholded likelihoods from intensity-discontinuity detection. $\alpha$ and $\beta$ as in Fig. 7 b) Depth reconstruction. c) Depth discontinuities.

particular, if the depth observations are less reliable than the intensity observations (i.e. they have larger variation from their expected true values), the line sites tend to have larger (negative) stability measures than the depth sites at the early stage of the computation. This means the line sites commit (since they are based on intensity information) earlier than the depth sites, resulting in a final configuration that is more consistent with the intensity discontinuities. At the locations where depth observations are missing (e.g. Figs 5, 6, 8), the intensity discontinuity information helps to localize the depth discontinuities before the depth information can be spatially propagated. Similarly, when the depth observations are more reliable (denser, less noisy), the depth sites commit earlier. The early depth commitments influence the stability measures, and thus the later commitments, of the line sites. The resulting configuration tends to be more consistent with the depth data.

*The Computational Advantages of HCF*: The performance characteristics of HCF in minimizing the energy of coupled MRFs is consistent with its performance on simple MRFs, incorporating only the line process, reported earlier [Chou and Raman 1987]. That is, the enhanced HCF algorithm behaves efficiently and predictably. The introduction of the continuous-valued depth process requires more visits to the depth sites than occurred in the optimization using only the binary line process. The line-process only MRF stabilized after fewer than 1.01 visits per site (on the average) [Chou and Raman 1987]. Experiments with the checker board images (Figs 7), show that it takes on average fewer than 3 visits per site to achieve reasonable estimates in the coupled intensity-depth MRFs. The situation is complicated by the fact that the sizes of the regions affect the speed of convergence: Larger regions require on average more visits per site for results to propagate through them.

Since the energy functional is quadratic given a line configuration, in principle any deterministic minimization method would find the same minimal configuration of the depth process. However, there are some advantages to HCF over iterative relaxation schemes with predetermined visiting orders. HCF always visits the site that can reduce the energy measure the most. Thus early visits are far more important than the later ones with HCF. The rate of stabilizing is always maximized, and the most reliable decisions, which reduce energy most, are made first, and at some point the computation may be terminated with confidence of negligible future improvement. Fixed-order schemes cannot guarantee this property.

### 4.2. Future Work

We plan to continue to explore the properties of the HCF algorithm and the extensions necessary to deal with sparse, continuous labels in MRFs. The following activities are ongoing and orthogonal, so they can be integrated in any combination as results accrue.

(1) Use real data. The Cooper stereo algorithm [Cooper 1987] yields sparse disparity data associated with intensity contours in the input images. It has been demonstrated to be robust and to work with natural scenes and with structured light. We plan to use this algorithm, or perhaps others under development that may yield denser data, to evaluate the performance of the multimodal segmentation and reconstruction system when the input is the results of other computer vision processes.

(2) Parallel HCF implementation. The HCF algorithm relies on a priority queue of all MRF elements in order of stability. A heaping algorithm is an efficient implementation in a serial model of computation, but we are exploring parallel alternatives both for reasons of speed and because the problem is intrinsically interesting.

(3) Distributed asynchronous data sources. The HCF algorithm so far has only been tested under the condition that it is presented with all the data simultaneously, and thus can correctly find the globally highest-confidence element. Under different circumstances, the data may arrive partially or asynchronously. The question then is how is HCF performance is affected by data arrival that may result in spatial or temporal discontinuities in the MRF element field.

(4) Spatially nonuniform MRFs and serial data accumulation. We are interested in fields with a peripheral and foveal organization, with a central high resolution area surrounded by a (perhaps progressively) low-resolution one. Call such an organization a retina. Over time, "eyemovements" can reposition the retinal organization within a larger, uniformly high-resolution MRF representing a stable world, or its projection. The interaction of the elements over time is of interest. If an element is seen first in the periphery and is then foveated, it proceeds through a "coarse to fine" context that may allow it to reach a correct labelling more reliably. There are several technical questions about the implementation of this idea.

(5) Computational advantages. With HCF one can set a threshold on stability measures that will terminate the computation with high confidence that only insignificant changes of the depth configuration would occur if the computation would continue. This property is bought at the cost of maintaining the priority order HCF needs, and raises the obvious question: "Is the gain worth the cost?". We are studying this tradeoff between the overhead paid in deciding the dynamic visiting order of HCF and its computational gain of fewer visits to the sites.


### Acknowledgements

We thank Dave Sher, Rajeev Raman, Myra Van Inwegen, Lata Narayanan, and Paul Turner for their contributions to our understanding and to the simulator software. We would also like to thank Tomaso Poggio and the members of Rochester Vision Group for their valuable suggestions and encouragement.

This work was supported in part by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Bolling AFB, DC 20332 under Contract No. F30602-85-C-0008. This contract supports the Northeast Artificial Intelligence Consortium (NAIC). This work was also supported in part by U.S.Army Engineering Topographic Laboratories research contract no. DACA76-85-C-0001, in part by NSF Coordinated Experimental Research grant no. DCR-8320136, in part by ONR/DARPA research contract no. N00014-82-K-0193, and in part by a grant from the Eastman Kodak Company. We thank the Xerox Corporation University Grants Program for providing equipment used in the preparation of this paper.


### References


Blake, A. and A. Zisserman, *Visual Reconstruction*, MIT Press, 1987.

Brown, C. M., "PADL-2: A Technical Summary", *IEEE Computer Graphics and Applications*, Mar. 1982, 69-84.

Chou, P. B. and C. M. Brown, "Probabilistic Information Fusion for Multi-Modal Image Segmentation", *Proceedings: IJCAI-87*, Milan, Italy, Aug. 1987.

Chou, P. B., C. M. Brown, and R. Raman, "A Confidence-Based

Approach to the Labeling Problem", *Proceedings: IEEE Workshop on Computer Vision*, Miami Beach, Florida, Nov. 1987, 51-56.

Chou, P. B. and R. Raman, "On Relaxation Methods Based on Markov Random Fields", TR 212, Computer Science Dept., The Univ. of Rochester, July 1987.

Cooper, P. R., "Order and Structure in Correspondence by Dynamic Programming", TR 216, University of Rochester, June 1987.

Gamble, E. and T. Poggio, "Visual Integration and Detection of Discontinuities: The Key Role of Intensity Edges", MIT A.I. Memo No. 970, October 1987.

Geman, S. and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-6*, 6 (Nov. 1984), 721-741.

Grimson, "Binocular Shading and Visual Surface Reconstruction", *CVGIP*, 1984, 19-44.

Horn, B. K P., "Obtaining shape from shading information", in *The Psychology of Computer Vision*, Winston, P. H. (editor), McGraw-Hill, New York, 1975, 115-155.

Marr, D., *VISION*, W. H. Freeman and Company, 1982.

Marroquin, J. L., "Probabilistic Solution of Inverse Problems", AI-TR 860, MIT Artificial Intelligence Laboratory, Sep. 1985.

Marroquin, J., S. Mitter, and T. Poggio, "Probabilistic Solution of Ill-Posed Problems in Computational Vision", *Proceedings: Image Understanding Workshop*, Dec. 1985, 293-309.

Poggio, T., "Integrating vision modules with coupled MRFs", Working Paper No. 285, MIT AI Laboratory, 1985.

Sher, D. B., "A Probabilistic Approach to Low-Level Vision", TR 232, University of Rochester, Oct. 1987.

Stuth, B. H., D. H. Ballard, and C. M. Brown, "Boundary conditions in multiple intrinsic images", *Proceedings: 8th International Joint Conference on Artificial Intelligence*, 1983, 1068-1072.

Terzopoulos, D., "Multilevel computational processes for visible surface reconstrucion", *Computer Vision, Graphics, and Image Processing 24* (1983), 52-96.

Terzopoulos, D., "Integrating Visual Information from Multiple Sources", in *From Pixels to Predicates*, Pentland, A. P. (editor), Ablex Publishing Corp., 1986, 111-142.

Torre, V. and T. Poggio, "On Edge Detection", *IEEE Transactions on Patter Analysis and Machine Intelligence PAMI-8*, NO. 2 (Mar. 1986), 147-163.

# IU AT UI: AN OVERVIEW AND AN EXAMPLE ON SHAPE FROM TEXTURE

Narendra Ahuja
Thomas Huang

*Coordinated Science Laboratory*
*University of Illinois*
*1101 West Springfield Avenue*
*Urbana, Illinois 61801*

This paper is in two parts. The first part presents an overview of some recent research in image understanding (IU) at the University of Illinois (UI). This includes an overview of the integration approach which we have used in a number of IU problems. Several examples of our research that use integration for image interpretation are summarized. The second part describes in detail the use of integration in solving the shape from texture problem.

## 1. OVERVIEW OF RECENT IU RESEARCH AT UI

A major part of our recent research is in four different areas of image understanding. The first area (Sec. 1.1) deals with integration of multiple image cues in performing image interpretation. These cues capture different aspects of the scene structure, and their integrated analysis leads to a more robust inference about the scene characteristics than possible from individual cues. The second area (Sec. 1.2) is concerned with our work on interpretation of image sequences showing dynamic scenes. Here we estimate the three-dimensional (3-D) motion parameters of moving objects and the 3-D surface structure, given feature correspondences over a sequence of images. We develop compact models of scene motion that can make predictions about future scenes. Projects in the third area (Sec. 1.3) report work on different components of an evolving 3-D representation and navigation system. The goal here is to build a system that can acquire, maintain and use 3-D information about its environment. Since the emphasis is on a working system, computational complexity and operating speed are of central importance in our work in this area. Finally, in the fourth area (Sec. 1.4), we summarize our work on a specific multiprocessor architecture that we have proposed for image understanding computations. The architecture is designed to efficiently execute divide-and-conquer algorithms for image computations. Some of the projects in these areas are summarized in the following subsections. To keep the paper brief, we have not included in the overview section (Section 1) any discussion of and references to relevant work done by others; such discussion and references are available in our publications cited. However, a comparitive analysis with and references to other approaches are contained in Section 2, which presents a detailed description of one of our recent integration examples.

### 1.1. Integration

Our goal in this area is to perform 3-D or other interpretation of images, such that the interpretation simultaneously satisfies a range of constraints imposed by the image structure and the model of the scene. We use a simple set of processes each having a narrow area of expertise. The processes carry complementary or redundant information obtained from different image cues. Image interpretation is derived as a result of a cooperative computation that resolves conflicts and ambiguities arising from the individual processes. We summarize below examples of the integration approach on three specific problems: surfaces from stereo, surfaces from focus, stereo and vergence, and perceptual grouping in dot patterns. A fourth example of integration, on recovery of the orientation of textured surfaces from image texture, is presented in detail in Section 2.

#### 1.1.1. Integrating Stereo Matching, Contour Detection and Disparity Smoothness

The purpose of stereo algorithms is to take two images of a scene, from slightly different viewpoints, and produce a complete depth map of the visible surfaces. The *usual* paradigm of these algorithms is: (1) *detect* suitable features in each image, (2) *match* corresponding features to determine their depths, and (3) *interpolate* to obtain a complete depth map. For the most part, the features that have been used in stereo have been low level. However, due to their simplicity, low level features can have many ambiguous matches, and this makes the matching step difficult. Also, occluding and ridge contours in the scene (where the depth and orientation, respectively, change abruptly) create difficulties for matching and interpolation.

Algorithms developed in the past complete the matching process *before* interpolating to obtain a dense depth map. Uniqueness of matching is only enforced by conditions that involve simple local relationships among disparity values as mentioned above and *not* the properties of the resulting surface. However, since a given disparity value implies depth a value, a stereo pair with matching ambiguities implies multiple surfaces, having different smoothness properties. The relative acceptability of these surfaces should be determined by the nature of the real world objects, namely. that their surfaces are smooth in the sense that the normal direction varies slowly, except across relatively rare ridges. Since, effectively, the matching process determines the final surface derived, the correctness of the choice of matches ought to be judged by the type of surfaces it produces. Therefore, the interpolation process should be used to choose correct matches. The matching and interpolation processes thus should be

integrated. This is in contrast to the traditional *first-finish-matching-then-interpolate* approach used by stereo algorithms developed in the past.

Through a pilot study, we have tested the merit of of our *smoothness-of-disparity* constraint in human stereo vision, against the *constant-local-disparity* constraint used in the past (Hoff and Ahuja, 1985, 1987). We have developed an approach that integrates the processes of feature matching, contour detection, and surface (disparity) interpolation. Integration is necessary to ensure that the detected surface is smooth. The surface interpolation process takes into account the detected occluding and ridge contours in the scene; interpolation is performed within regions enclosed by these contours. Planar and quadratic patches are used as local models of the surface. Occluded regions in the image are identified and are not used for matching and interpolation. The approach developed is fairly domain-independent since it uses no constraint other than the assumption of piecewise smoothness. A coarse-to-fine algorithm is used that requires no human intervention other than an initial rough estimate of depth. The surface estimate obtained at any given level of resolution is used to predict the expected locations of the matches at the next finer level. As the final result, a multiresolution hierarchy of surface maps is generated, one at each level of resolution.

The most characteristic and novel feature of our approach is the use of the surface smoothness criterion for stereo matching, and thus an integration of matching and surface interpolation operations. The control passes back and forth between matching and interpolation processes, each depending on the result of the other to make progress, and generating a progressively refined set of depth maps of a scene at increasing degree of resolution. A given coarse level surface predicts the locations of edge matches at the next finer level. The matched features at the finer level provide a more refined surface which in turn predicts pairs of edges to be matched at the next finer level of resolution. Another unique characteristic of our approach is that our smoothness constraint explicitly incorporates the existence of depth and orientation discontinuities in the computation. It is domain independent, i.e., it uses no world knowledge other than the constraint that objects in the real world tend to have smooth surfaces, i.e., the depth/orientation varies gradually except across relatively rare, occluding/ridge contours. In our approach, these contours are *constantly* detected and a smooth surface is interpolated allowing depth/orientation discontinuity across the contours, thus implementing the *piecewise-smooth* model of real world objects. Finally, our approach enforces smoothness and continuity of 3-D occluding and ridge contours. This constraint is based upon the assumption that real world objects have surfaces that have smooth borders.

A detailed discussion of the algorithm, performance and advantages that accrue from the use of the above integration approach can be found in (Hoff & Ahuja, 1987a,b). An early report on this work appears in the 1985 proceedings of the IU workshop (Hoff & Ahuja, 1985). Some experimental results indicative of the performance of the algorithm are shown in Figure 1.

### 1.1.2. Integrating Focus, Stereo and Vergence

Most work on stereo (ours included) has been concerned with the estimation of a *local* depth map, i.e., a depth map for the part of the visual field in the immediate vicinity of a reference. This is because only a limited part of the visual field around the reference point may be in sharp focus, or visible, at a time. Stereo algorithms require an initial, coarse estimate of the local surface through the reference point. The estimate must not be too far out of this range in order for the algorithms to succeed. (In our algorithm summarized above, this estimate is given as a frontal surface at a depth between the depths of the closest and the farthest objects in the small visual field. The algorithms then obtain 3-D structure over the small visual field.

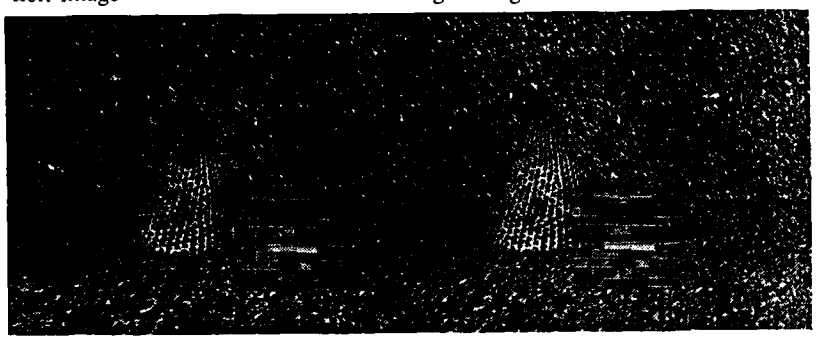
Real scenes are often large, and have large depth ranges. Therefore, camera directions and other imaging parameters need to be changed to photograph different parts of the scene. Like human eyes, the cameras must pan and tilt, converge and diverge, focus on near and far objects, to acquire the stereo images for different fixation points. Therefore, to reconstruct depth map from each stereo pair, a different initial estimate appropriate for the local surface must be given, which may be obtained from associated imaging parameters. Thus, a close interaction is necessary between the imaging parameters used and the stereopsis process, and hence the need for acquiring images in an integrated fashion with generation of depth map.

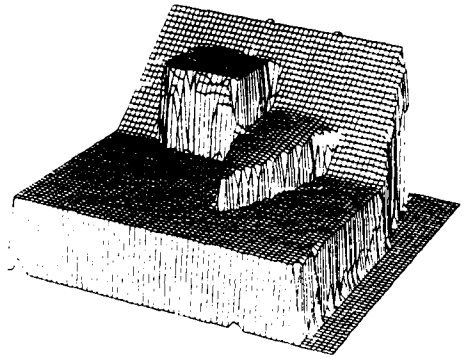
Several researchers have discussed the roles of focussing and camera vergence as sources of 3-D information, and have pointed out their computational as well as the biological significance. However, there has been only a limited use made of these sources in computational approaches, especially in a mutually cooperative mode. Focus, vergence and stereo disparity each has its strengths and weaknesses as a source of 3-D information. Focus provides an estimate of depth whose accuracy decreases with increasing object depth. Overall, focus provides a coarse depth estimate. Vergence provides a depth estimate of the point of intersection of the optical axes of the cameras which is fairly accurate for low vergence angles but whose accuracy decreases for objects far away. However, to use vergence it must be ensured that the two cameras actually are fixated on the scene point common to their optical axes. This may not be the case if the view of one of the cameras is obstructed by an object which does not intersect the optical axis of the other camera. Stereo disparity, of course, provides an accurate depth estimate if a coarse estimate is available initially. It is interesting to note that these strengths and weaknesses are complementary. For example, while stereo disparity can provide accurate surface reconstruction, it requires a coarse initial surface estimate which can be readily provided by focus and/or vergence. However, depth estimate of a scene point from vergence is valid only if it is ensured that the cameras are fixated at that point. For relatively close objects, this can be done by ensuring that the depth estimates for the image centers provided by the focus process are for the same 3-D point. For distant objects, this problem of possible occlusion from one viewpoint is less serious. Another problem that arises when stereo alone is used is that, in any given configuration, surface estimate can be obtained for only a limited part of the complete visual field of

Left Image

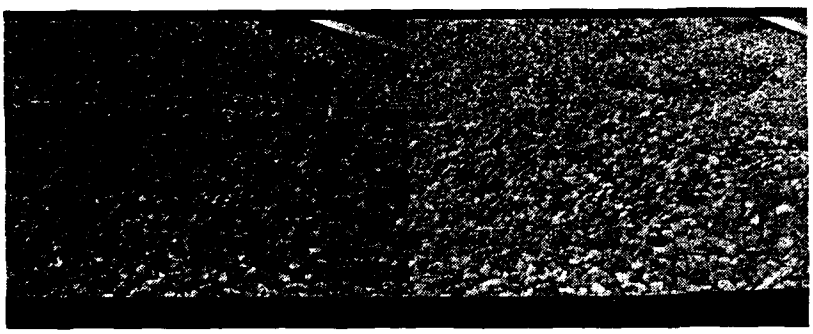
Right Image

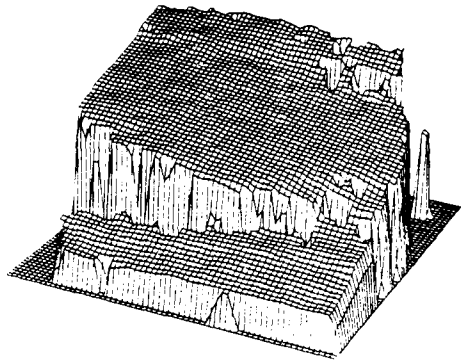
Reconstructed Disparity Surface



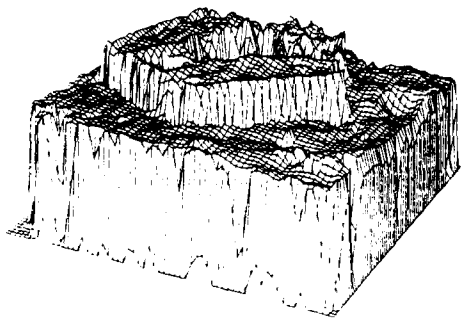


Synthetic image of a cone and a cube

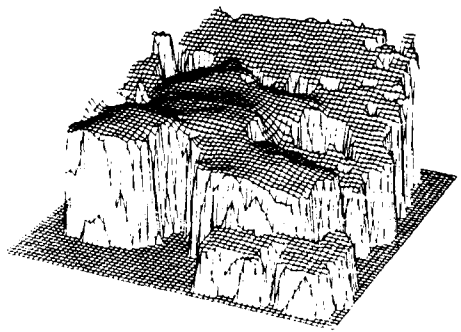




Real image of a mound of rocks



Real image of the Pentagon



Real image of Renault auto part

Figure 1. Integration of feature matching, contour detection and disparity smoothness in stereo vision.

interest. The part of the visual field imaged is limited both along the lateral dimensions and along the depth dimension. The former limitation occurs because of the limited field of view of the cameras, and may be remedied by changing the orientations of the two cameras so that their optical axes intersect different parts of the visual field. The latter limitation arises for two reasons. First, the entire surface may not be in focus simultaneously over its large depth range. Second, the entire surface may not give disparity values in a workable range; for example, the parts of the surface much closer than the point of fixation may give disparity values on the order of image dimensions, whereas those parts much farther away may give disparities that are too small (less than a pixel). This problem may be remedied by obtaining depth estimates of small parts of the surfaces at a time, having small depth ranges. The necessary stereo pairs of images of local surface patches can be obtained by changing the vergence angle of the cameras so the point of fixation moves along the depth dimension, while simultaneously adjusting focus to obtain sharp images. To reconstruct depth map from each stereo pair, a different initial estimate of the surface must be given which approximates the local surface in the vicinity of the fixation point. A computed local depth map may be extrapolated to predict vergence and focus parameters necessary for imaging adjacent parts of the scene. As the scan of the scene continues, depth maps generated for visual subfields around different fixation points must be merged to generate the depth map of the entire visual field, possibly having a much larger global depth range than the individual local maps.

We have developed an algorithm for achieving the integration described above. The current implementation makes limited use of camera vergence, and emphasizes the integration of focus and stereo. The algorithm repeats the following steps until the entire scene has been mapped: 1. Choose direction of gaze; 2. Fixate: repeat until both image centers are in focus at the same distance (2.1. vary focus to estimate depth at image centers, 2.2. Rotate camera platform and verge so that both cameras are aimed at nearest scene point); 3. Vary focus to produce a grid of depth estimates over the entire images; 4. Invoke stereo, supplying the depth estimates from focus; and, 5. Merge the resulting depth map with an evolving global scene description

There are several advantageous features of integraton that the current algorithm does not still incorporate. However, the current simple implementation does demonstrate the power of integration in dramatically improving the performance of surface estimation, over what would be possible if the individual depth cues were used independently. In particular, the algorithm obtains good surface maps of a large scene having a large depth range, as illustrated in Figure 2.

### 1.1.3. Integrating Region, Border and Component Gestalt in Extracting Perceptual Structure in Dot Patterns

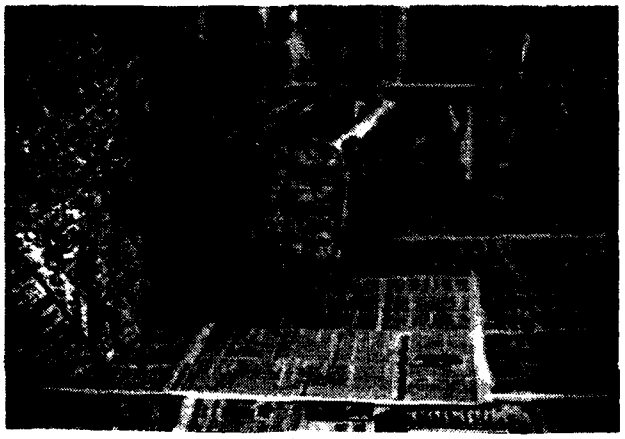
This research concerns perceptual grouping, or goal independent detection of perceptual organization in images. The organization, of course, is often at a range of scales. The image entities, or tokens, that may be grouped include blobs, edge segments, and geometrical features of image regions. These tokens have properties such as position, shape, size, orientation, color, brightness, and the termination points (if the tokens are elongated or curvilinear). The roles of some of the properties in grouping may be complex. To reduce this complexity, a first step toward understanding the grouping phenomenon may be to study the roles of some relatively simple properties. One way of accomplishing this is to eliminate all but one property at a time and examine the effects of that property on grouping. Since dots are without size, orientation, color and shape, dot patterns provide a means for studying the effect of token positions on their grouping, while minimizing, the role of nonpositional properties. We call such initial grouping of dots based only on their positions as the lowest level grouping.

The single variable that determines the grouping of dots is the relative locations, or proximity, of dots. Four possible perceived structures in dot patterns are: (a) a cluster with nonempty interior, (b) a cluster with no interior (e.g., a bar), (c) curvilinear structures, and (d) a single-dot cluster. In the first case a dot may lie either along the border of a cluster or in the interior region. The goal of the computation of the lowest level groupings may be achieved by assigning to each dot one of the above roles, or one of the following labels: *interior, border, curve* and *isolated.* We have developed a computational approach to perceptual grouping in dot patterns. Central to our model of grouping is the issue of representation. How should the geometric structure of a dot pattern be represented? How is the perceptual structure related to the geometric structure? How are local and global perceptual organizations related? How should the perceptual structure be estimated? How should the estimates be checked for their validity? How should the differences between the model's predictions and the human segmentation be used to improve the computational model? Our work incorporates one set of answers to some of these questions in the defining a computational approach.

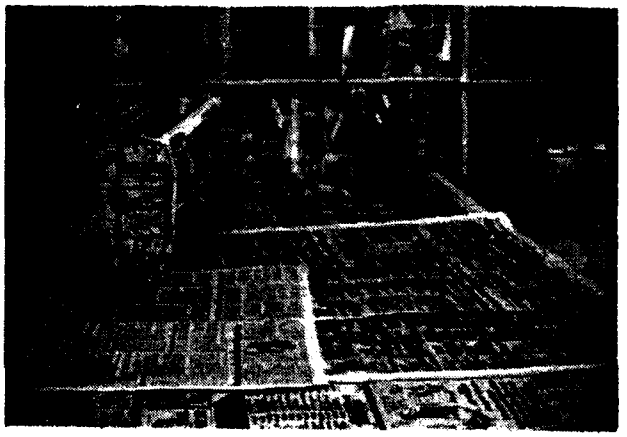
To perform the lowest level grouping, first the geometric structure of the dot pattern is represented in terms of certain geometric properties of the Voronoi neighborhoods of the dots. (The Voronoi neighborhoods of dots are defined in terms of the Voronoi tessellation of the plane generated by the dot pattern. The Voronoi tessellation is a partition of the plane where the edges are the bisectors of nearby dots. Each cell of the tessellation contains exactly one dot, and defines the neighborhood of that dot. This definition of the neighborhood of a dot captures the intuitive notion of neighborhood in many ways (Ahuja, 1982). The properties of the Voronoi neighborhoods are then related to the primitives of the perceptual structure, e.g. interiors of blobs, borders of blobs, curves and isolated dots. The association between the perceptual roles of dots and the geometric structures of their Voronoi neighborhoods, however, are only *typical.* The presence of a certain local geometric structure is neither necessary nor sufficient for a certain perceptual interpretation of a given dot - the local geometric structures of other dots, near and far, and their perceptual interpretations have a profound and usually domineering influence on how the given dot is perceived. "Faults" in the expected local geometric structures of a limited number of dots are tolerated in favor of Gestalt properties such as smoothness of borders or curves, or compactness of com-

(a) View direction 1, left image.

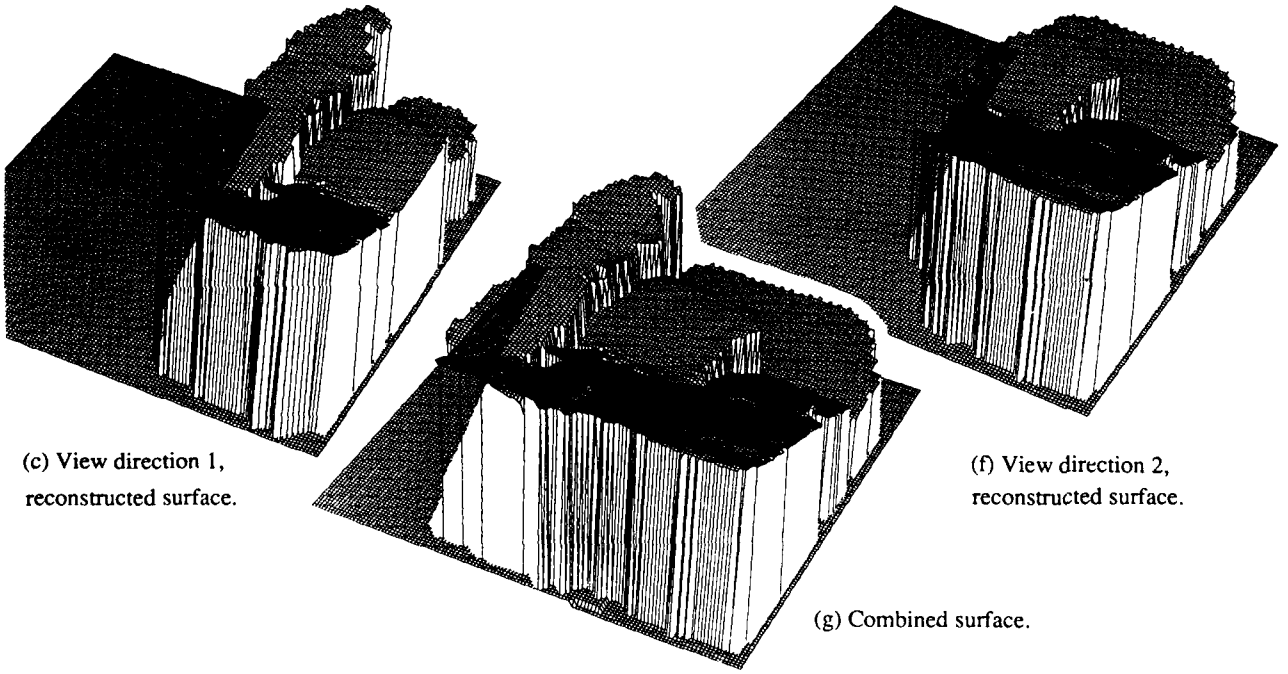
(d) View direction 2, left image.



(b) View direction 1, right image.



(e) View direction 2, right image.



(c) View direction 1, reconstructed surface.

(f) View direction 2, reconstructed surface.

(g) Combined surface.

Figure 2: Integration of focus, stereo and vergence. (Vergence is used to a limited extent in the current implementation.)

ponents, thus resulting in global interpretations that may assign such perceptual roles to dots which conflict with their local geometric structures. The grouping is seeded by assigning to dots their locally evident perceptual roles. This is done through independent modules that possess narrow expertise for recognition of typical interior dots, border dots, curve dots and isolated dots, from the properties of the Voronoi neighborhoods. The results of the modules are allowed to influence and change each other so as to result in perceptual components that satisfy global, Gestalt criteria. Thus, an *integration* is performed of multiple constraints, active at different perceptual levels and having different spatial scopes in the dot pattern, to infer the lowest level perceptual structure. The constraints that are integrated include local structure of dots, smoothness of borders, and compactness of components. Starting with the first constraint on the local structure, the following two constraints incorporate information having increasingly global spatial scope in the dot pattern. The local interpretations as well as lateral corrections are performed through constraint propagation using a probabilistic relaxation process. The result of the lowest level grouping phase is the partitioning of a dot pattern into different perceptual segments or tokens.

In the implementation of the approach, we have attempted to minimize the use of *ad hoc* thresholds for decision making. Instead, we have tried to increase the amount of information used, whenever possible, to reduce any ambiguity in interpretation.

The lowest level grouping constitutes an important first step in extracting perceptual structure in dot patterns. Subsequent steps would further group the lowest level tokens to identify any hierarchical structure present. The grouping among tokens is again done based on a variety of constraints including their proximity, orientations, sizes, and terminations, *integrated* so as to mimic the perceptual roles of these criteria. The result of the gouping of lowest level tokens is even large tokens. The hierarchical grouping process repeats until no new groupings are formed. The final result is a representation of the heirarchical perceptual structure in a dot pattern. Figure 3 shows some dot patterns, along with the lowest level structures found by our algorithm. Also shown is an example of the hierarchical grouping found. Details of this work are reported in (Ahuja and Tuceryan, 1987; Tuceryan & Ahuja, 1983, 1987).

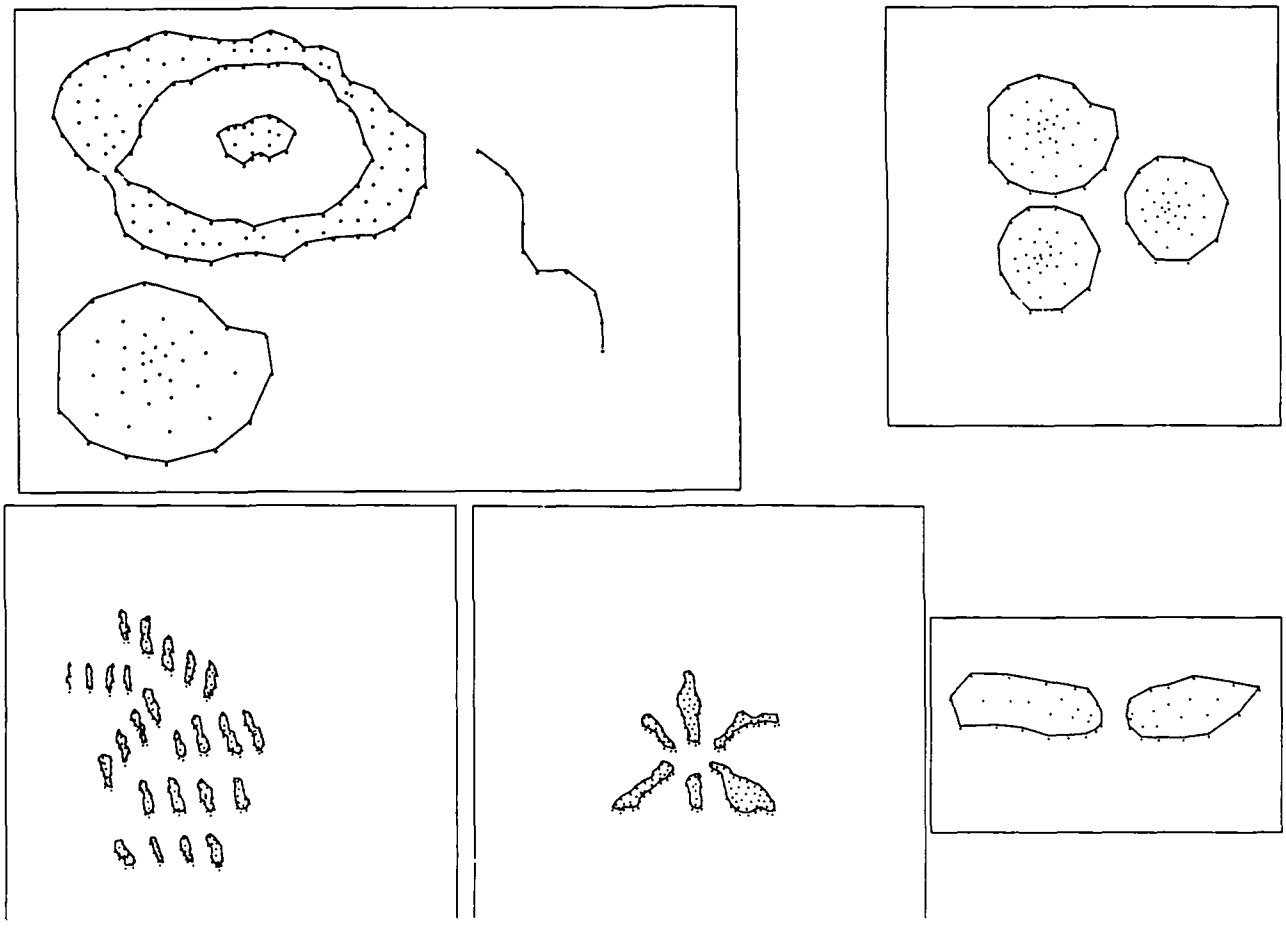


Figure 3. Integration of region, border and component Gestalt. The lowest level groupings of dots are shown by the detected borders.

## 1.2. Motion Analysis

The long-range goal of our research in this area is the understanding of dynamic scenes. A major problem in this direction is to estimate the scene surfaces as well as 3-D motion parameters from a sequence of images. The framework we have used consists of three stages: finding feature correspondences in a sequence of frames, determining motion parameters and surface structure from the correspondences, and finally, characterizing motion exhibited in the frame sequence.

In most of our work so far, we have assumed that the feature correspondences are given. We have done limited work on this first stage (Gu & Huang, 1985; Gu, Yang & Huang, 1987) which is highly scene dependent. In the following subsections, we will summarize our research results on the second and third stages.

### 1.2.1. Motion and structure from two images

Given a sequence of stereo images and feature corespondences, the 3-D positions of the features are determined by triangulation. The problem is then determining motion parameters from 3-D positions of the features. Though a closed-from solution for motion parameters is easy to derive, it is necessary to ascertain the impact of noise in the data on the results of estimation. We have developed an algorithm for determining least squares estimates of motion parameters in the presence of noise, and compared its performance with other algorithms (Arun, Huang & Blostein, 1987). We have found from extensive analysis that uncertainity in the estimates of the 3-D positions due to image sampling on the estimates of 3-D positions obtained by the stereo triangulation is larger in the forward direction (say, $Z$–direction) than in the $X$ and $Y$ directions (Blostein & Huang, 1987).

Given a monocular image sequence and feature correspondences, the problem of determining motion parameters and structure of the scene is harder than the case of stereo sequence because the 3-D position information is available in a more indirect form in the monocular image sequence. We developed a closed-form solution using point correspondences (Tsai & Huang, 1984). However this algorithm is primarily meant for a noise free case, and it is very sensitive to noise as reported in (Tsai & Huang, 1984) and (Fang & Huang, 1984). Further, for the case where the corresponding points lie along a planar patch, this algorithm fails. Therefore, we developed a new algorithm in which overdetermination of intermediate parameters is used to combat noise. A method to estimate the errors in the solution is also given. Specifically, the standard deviation of the error is calculated in terms of the variance of the errors in the image coordinates. Our experiments show that acceptable accuracy (under 10% relative errors for motion parameters) is achieved for 256 by 256 images. The details can be found in (Weng, Huang & Ahuja, 1987b). We have also developed algorithms that successfully analyze correspondence data from planar surfaces (Tsai & Huang, 1981; Tsai, Huang & Zhu, 1982; Weng, Ahuja & Huang, 1988b).

To further improve the robustness of these algorithms, we have developed a two step approach to motion parameter and struture estimation. The first step in this approach is to estimate the motion parameters using a robust linear algorithm that gives closed-form solution for motion parameters and scene structure. The second step is to optimize the results given by the linear algorithm using maximum likelihood estimation. The details of these two steps can be found in (Weng, Huang & Ahuja, 1987c) and (Weng, Ahuja and Huang, 1988a). We have observed that the stability of motion parameters strongly depends on the type of motion and system parameters. The field of view of a sensor, magnitude of translation, and direction of translation are among the parameters that significantly affect the accuracy of the solutions. We have studied qualitative relationships among these parameters, and empirically demonstrated these relationships though a series of simulations (Weng, Huang & Ahuja, 1988b).

The choice of the type of features depends on their availability in the images and the reliability of their measurements. When points are not available in large quantities, other features such as lines can be used. Since the higher level features such as lines and edges are determined by a set of pixels, the redundancy in the pixels make it possible to locate those features accurately. We developed algorithms that use line correspondences to solve for the motion parameters (Yen & Huang, 1983; Liu & Huang, 1986). These algorithms are iterative, and like many iterative algorithms, they do not guarantee a solution, and exhibit great sensitivity to noise. To avoid this problem, we have developed a closed-form solution to motion and structure from line correspondences from monocular perspective image sequences. The algorithm requires a minimum of 13 lines over three perspective views. A unique solution to motion and structure is guaranteed if and only if the line configuration is not degenerate and the translation between any two views does not vanish. Necessary and sufficient conditions for degenerate spatial line configurations have been derived. The algorithm uses weighted linear least-squares techniques to combat noise and so is more robust. Details can be found in (Weng, Huang & Ahuja, 1988a).

### 1.2.2. Motion modeling and prediction

The algorithms discussed above estimate structure and motion parameters from subsequences of images, without attempting to develop a compact model of the scene that might unify the estimated parameters. If such a model were extracted, it would make it possible, for example, to interpolate motion and supply any missing image frames or to correct any erroneous frames, or to extrapolate motion to predict future positions of moving objects. Human vision is adept at using image sequences to quickly develop an understanding of the scene required for such capabilities. For example, after a football is kicked off, people can judge whether the football will pass through uprights long before it actually reaches there.

A natural approach to building a compact model of the scene motion is to use principles of dynamics that govern 3-D object motion. We have introduced a locally constant angular momentum (LCAM) model. The model is local in the sense that it is applied to a limited number of image frames at a time. Specifically, the model constrains the motion over a local frame subsequence to be a superposition of precession

and translation. Thus, the instantaneous rotation axis of the object is allowed to change through the subsequence. The trajectory of the rotation center is approximated by a vector polynomial. The parameters of the model evolve in time so that they can adapt to long term changes in motion characteristics (Weng, Huang & Ahuja, 1987a).

The nature and parameters of short term motion can be estimated continuously with the goal of understanding motion through the image sequence. We have developed such an estimation algorithm. The algorithm is linear, *i.e.*, it consists of solving simultaneous linear equations. Based on the assumption that the motion is smooth, object positions and motion in the near future can be predicted, and short missing subsequences can be recovered. Figure 4 shows a sequence of stereo images taken from a model airplane. Starting from the fourth frame, the prediction is made for the 3-D coordinates of the feature points at the next time instant. The relative maximum prediction errors are shown to the left of the corresponding frame pairs in Figure 4.

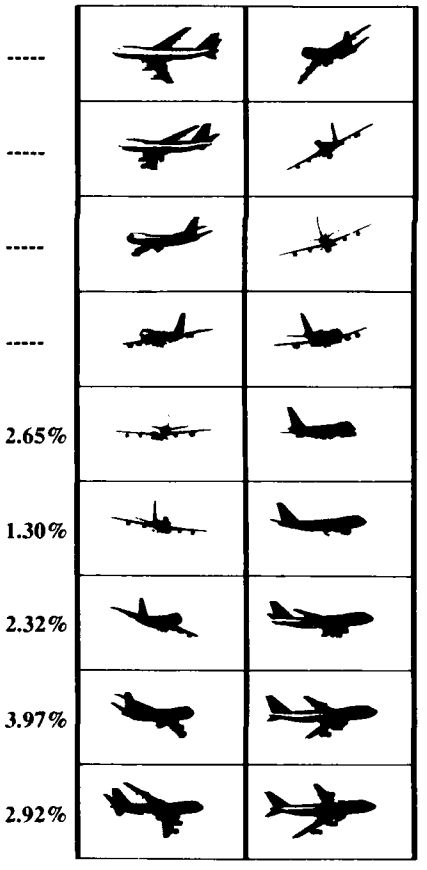



Figure 4. Image frame sequence of a model airplane:

left column : left view; right column : right view.
(beginning with the fifth frame, the maximum relative prediction errors are shown to the left of the corresponding frame pairs.)

### 1.2.3. Motion of multiple objects

In Sections 1.2.1 and 1.2.2, we are concerned with a single rigid object in relative motion with respect to the camera. In many situations, the scenes contain multiple objects moving differently. Therefore, one faces a segmentation problem along with the problem of motion/structure estimation. With stereo sequences, rigidity constraints can be used to to do the segmentation. Along this line, we have developed algorithms for matching 3-D features which can be used to estimate motions of multiple objects (Chen & Huang, 1987).

### 1.2.4. Nonrigid objects

Many scenes of interest contain nonrigid objects, e.g., people. We have developed an algorithm for analyzing the motion of objects consisting of jointed rigid parts (e.g., robot arms). The algorithm is based on the use of Gaussian and mean curvatures of the surfaces of the objects (Goldgof, Huang & Lee, 1988).

## 1.3. Deriving 3-D Occupancy Maps from Images, and Navigation

The goal of our work in this area is to develop algorithms and systems for generating maps of the occupancy of 3-D space by objects, and for deriving collision free and efficient trajectories to move an object from a given source location/orientation to a given destination location/orientation through an environment with a given occupancy map. We are concerned with only a coarse 3-D representation of filled/empty space, not with representing fine shape details of occupied space. There are three different aspects to representation of spatial occupancy: initial *generation* of the representation for a given scene, its *maintenance* as the objects in the scene move, and its *use* in environment manipulation including tasks such as planning trajectories of moving objects through the scene. Our work has addressed all three of the these aspects. We have used a single representation, the *octree*, in all three parts . This has helped integrate our efforts in the different parts, so that they have contributed to an evolving system. These three parts are described in the following subsections.

### 1.3.1. Octree generation from orthographic views

We have developed an efficient algorithm that derives the octree representation of an object, given a sequence of object silhouettes as obtained from any subset of 13 different vantage points. These viewing directions include 3 directions parallel to the three axes, and 6 directions parallel to the bisectors of angles, two each between x and y, y and z, and z and x pairs of axes. These correspond to the three "face-on" and the six "edge-on" directions of viewing an upright cube centered at the origin. The remaining 4 directions correspond to the four "corner-on" views of the upright cube. The octree is generated by effectively projecting each silhouette as a cylinder onto the octree space, and identifying the nodes whose corresponding cubes fall completely within or outside the cylinder. Successive silhouettes generate cylinders with different orientations and cross sections, and their evolving intersection provides an increasingly accurate representation of the object. Projection of a silhouette onto octree space is actu-

ally accomplished by a recursive subdivision of the silhouette image into quadrants and extending squares into cylinders, instead of working with cylinders having cross sections as complex as the object silhouettes. As a consequence, detection of intersection of a cylinder with the octree space amounts to a *table look-up* operation. This simplicity is afforded by the restriction of views to the 13 mentioned above, and the resulting fixed relationships between quadrants of the silhouette image and octants of the octree space, which eliminates the time spent in computing intersections. We have empirically measured the average *performance* of our system from the octrees generated for a selected set of objects. The measure of accuracy for a given object and orientation is the ratio of the volume of the object to the volume of the intersection of the extended silhouettes of the object. To obtain the average performance a Monte Carlo simulation experiment is performed involving a large number of executions of the following three step procedure. *First*, an arbitrary object from the chosen set and a random orientation is selected. *Second*, the object is projected along each viewing direction to provide a set of silhouette images. *Finally*, the octree is constructed and the corresponding object volume computed. The ratio of the actual to the computed volume is the desired result for the chosen object and orientation. Then, for a given set of objects, the measure of accuracy for a set of viewing directions is the estimated expected value of the ratio of the object volume to the constructed volume for a randomly selected object at a randomly selected orientation. We have made measurements on objects that are often used as primitives for 3-D representations, e.g., generalized cones, and some complex objects. Figure 5 shows a display of some objects as represented by the generated octrees. The details of the octree generation algorithm can be found in (Ahuja & Veenstra, 1988). A brief description of the algorithm appears in (Veenstra & Ahuja, 1986).

### 1.3.2. Octree generation from perspective views

The above algorithm accepts orthographic views. Thus, the objects are assumed to be far away from the camera(s). For the case when the images cannot be treated as orthographic, we have developed an efficient algorithm for generating octrees from multiple perspective views of an object. This algorithm does perform the necessary intersection between object and octree nodes. The intersection tests are designed to be computationally efficient. The algorithm first obtains a polygonal approximation of the object silhouette. This polygon is then decomposed (if necessary) into convex components. For each convex component, a pyramid is formed treating the view-point as its apex and the convex components as one of their cross-sections. The octree representation of each of these pyramids is obtained by performing intersection detection of the object with the cubes corresponding to octree nodes. The intersection detection step is made efficient by decomposing it into a coarse-to-fine sequence of intersection tests. These tests exploit the known cubical and pyramidal nature of the shapes to make intersection detection efficient, rather than perform a general three-dimensional polyhedral intersection detection. The first test approximates the pyramid by the smallest enclosing cone, and the cube by the smallest enclosing sphere. It is then easily determined if the sphere and the cone do not intersect, meaning that the octree node is white. If they intersect, a second test is performed to eliminate a few cases of no intersection. In this test, if all eight vertices of a cube are found to be contained inside the pyramid, it is decided that the cube and pyramid do not interstct; if some vertices are found to be inside and some outside, iit is concluded that they intersect. If all vertices are found to be outside, then the cube may or may not intersect with the pyramid. To resolve this problem, a third test examines if all eight vertices of the cube lie on the outside of the face, or, if not, whether the edges of the pyramid intersect with the extended faces of the cube. This gives the final set of octree nodes due to the pyramid. The octree for one silhouette is obtained by taking the union of octrees obtained for each component. Such octrees obtained from the different silhouettes are intersected to obtain the final octree of the object which represents the volume of intersection of the different pyramids. To measure the accuracy of the resulting octree, we use the ratio of the volume of the object reconstructed from the octree representation to the actual volume of the object, as for the case of orthographic views above. The algorithm and its performance on the same set of objects as used for the orthographic view algorithm above are reported in (Srivastava & Ahuja, 1987).

### 1.3.3. Octree updating for moving objects

A major feature of octree representation is the environment centered nature the spatial decomposition used. However, this same feature makes the representation sensitive to object motion. Thus, if a system is to use octrees to represent dynamic scenes, it must efficiently update octrees so as to track objects and keep the representation current. We developed a binary arithmetic that can be used to update octrees for translation of the represented objects. Each object node in the octree is identified by a sequence of edge labels encountered along the path to the node from the root node. Each successive label thus correspond to a cube of half of the size of the previous cube. The translation is expressed as a vector of three components, corresponding to three axial translations. Each component is expressed as a binary number. The location of each octree node after translation is obtained by finding its sequence after translation. The binary nature of the representations of the label sequence as well as the translation vector makes it possible to realize the translation through binary addition. Each leaf node is translated individually to obtain the updated octree. To obtain the new node after translation, the components of the desired translation vector are "added", one after another, to the above path vector according to our binary arithmetic. This algorithm is reported in (Ahuja & Nash, 1984). We further refined this algorithm so that it does not introduce new leaf nodes in the original tree corresponding to the interior of the object, since this would require compaction after translation. Translation is performed on nodes as high up in the tree as possible to minimize the computational effort. The details of this algorithm can be found in (Osse & Ahuja, 1984).

To allow for arbitrary rigid motion of the objects, we have developed an algorithm that updates the octree represen-

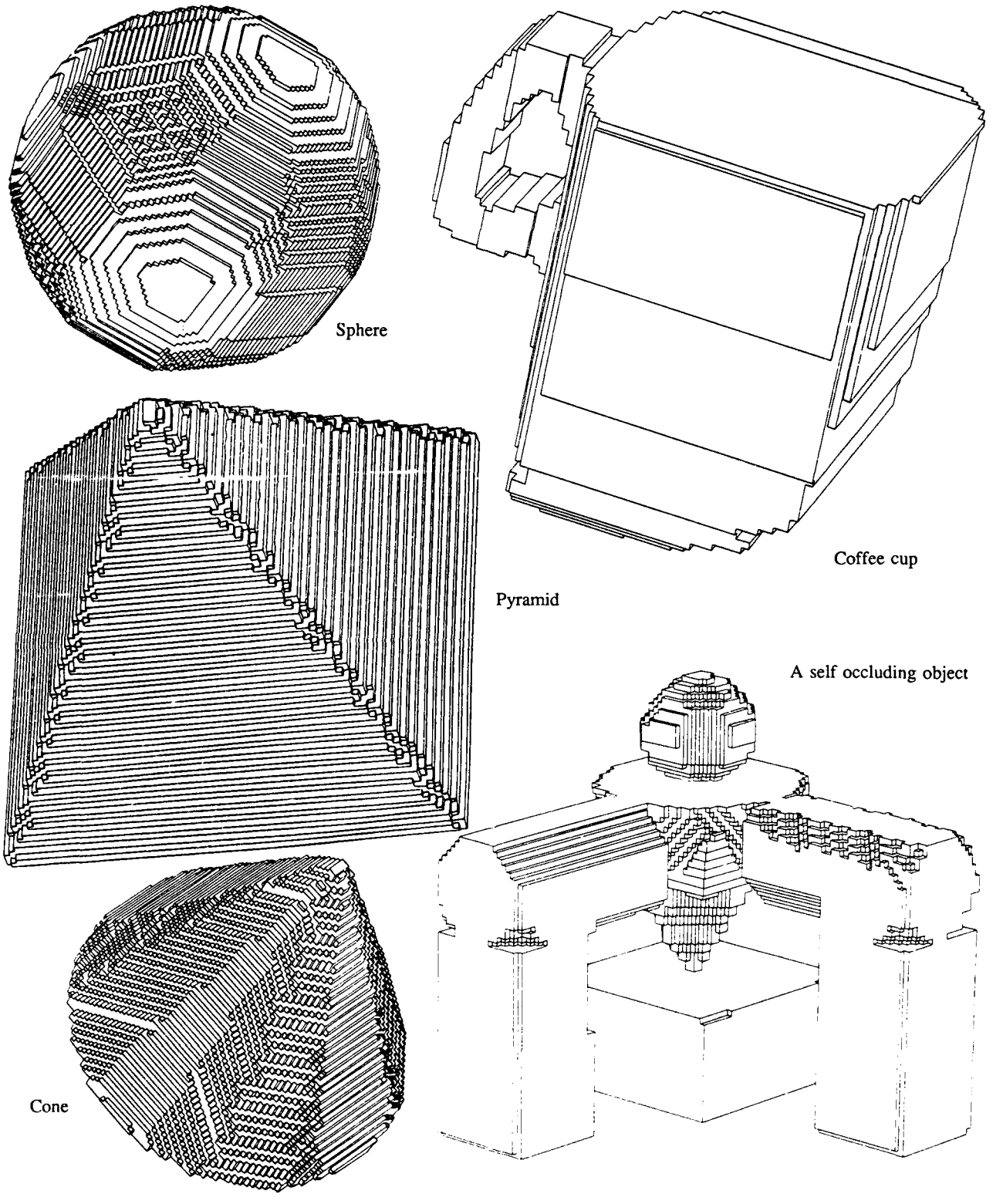



Figure 5. Line drawings displays of objects as represented by derived octrees.

tation of an object undergoing arbitrary translation and rotation. Because of the anisotropic nature of the primitives (cubes) used to define octrees, rotation of objects usually results in fragmentation or compaction of their octrees. Since some approximations must be made at the finest level of resolution to contain the octree size, the octree may undergo cumulative distortion as increasing number of rotations are performed. For example, a rotation by $\theta$ followed by a reverse rotation by $\theta$ may lead to a different octree. We have partly resolved this problem by keeping a compact octree representation of the object corresponding to a certain reference orientation. As the object is subjected to a sequence of rotations, a Euclidean sum of the rotation angles is maintained to indicate the precise current orientation. The current octree of an object is obtained by performing the appropriate rotation on the compact, reference octree. The efficiency of the algorithm accrues from efficient, coarse-to-fine test sequence used to determine the nodes of the new octree. The tests used are similar in spirit to those used in octree generation from perspective views. Details of this algorithm can be found in (Weng & Ahuja, 1987).

### 1.3.4. Display of octree represented objects

We have developed an algorithm to *display* the line drawing of the object represented by an octree. The display algorithm takes as input a pointer to the root of an octree and an arbitrary viewpoint, and produces as output a Tektronix compatible line drawing of the object with hidden lines removed. The algorithm makes use of the known spatial configuration of the cubes defining octree nodes in obtaining the sequence of lines to be displayed. The algorithm is a useful general purpose tool in working with octrees. For example, it can help monitor the progress made by the octree generation algorithm as successive views are assimilated, monitor octrees as they are updated for object motion, and monitor any other changes in octree structure. This algorithm was used to obtain the line drawings of objects presented in Sec. 1.3.1 to illustrate the performance of octree generation algorithm. Details of the display algorithm appear in (Veenstra & Ahuja, 1988).

### 1.3.5. Navigation

In this section, we summarize our wok on the use of 3-D representation, derivation and maintenance of which are accomplished by the algorithms described above. We have concentrated on the problem of path planning, i.e., deriving an efficient and collision free trajectory to move an object from a given source location/orientation to a given destination location/orientation through a given environment. A basic consideration that has guided our approach is the desired use of only local occupancy (obstacle) information in devising the local path of the object. This reduces the combinatorial complexity of the problem, while at the same time allowing the use of a global topological map to guide the choice of one among the available local, collision-free paths. To obtain a compact representation of free space, we have used a global potential field function. This representation is motivated by the electrostatic repulsion between like charges. Imagine that all the obstacles are composed of positively charged matter. If the moving object (MO) is also positively charged, the obstacle avoidance problem is resolved by the repulsive force. This force can be calculated as the negative gradient of a potential field. The potential field approach divides the problem into two stages. First, all possible paths between the starting and goal configurations are found, and a candidate path that is mostly likely to yield the shortest collision-free path for MO is selected. Second, three algorithms are used to modify the candidate path to derive the final path and orientations of MO.

The best candidate path is found as follows. First, a scalar potential field is computed due to all charged objects. Minima of the local potential field are identified. These minima capture the topological structure of the free space. All distinct paths between the starting and goal configurations are identified along the valleys of potential minima. The minimum potential valleys (MPV) represent possible paths for point objects and serve as initial estimates from which the final paths/orientations of MO are derived. The best candidate path is found from MPV using a cost function and dynamic programming. The cost function is based on the length of the paths and the width of the free space along the paths. The cost goes to infinity when the width of the free space is smaller than the width of MO, and decreases as the width of the free space becomes larger. After the cost is assigned to all the paths in MPV, dynamic programming is used to compute the shortest path with the minimum cost. The selected path serves as the initial estimate of the solution, and is used by findpath algorithms to derive the final path and orientation of MO. If the initially chosen candidate path does not yield a solution, the parts of the candidate path where MO collides with some of the obstacles are assigned infinite costs. Dynamic programming is run again to find the next best candidate path, and this path is used by the findpath algorithms.

The final paths are desired to be short and collision-free, and they should minimize the change in object orientation as it moves along the path. In problems where the free space between obstacles is wide, the final path and orientation of MO can be obtained by simply changing the initial candidate path away from the obstacles MO is colliding with. In other cases, the simple strategy of "running away" from the obstacles may not result in a collision-free path and orientations of MO. The initial estimate may have to be modified significantly to derive the final path, and the final path may require a complex change in the configuration of MO as it moves along the path. In still other cases, the initial estimates may even have to be changed topologically to obtain the final path. These three levels of difficulty of the findpath problem have led us to use three different optimization algorithms. Given an initial estimate, the final path can be derived by the parallel optimization algorithm (POA), or in more difficult cases by the serial optimization algorithm (SOA), or in the third case by the sidetracking algorithm (STA).

The parallel optimization algorithm employs a numerical method to minimize a weighted sum of the path length and the total potential experienced by MO along the path. The minimization of the potential pushes MO away from the obstacles, whereas the minimization of the path length keeps MO from wandering in wide parts of free space. This algorithm solves the easiest class of findpath problems. If some parts of

the free space are so narrow that MO must approach the narrow spaces with specific orientations, POA does not yield a collision-free path and orientations. The serial optimization algorithm moves MO along the candidate path, and identifies the narrow parts of the free space. SOA then finds collision-free configurations of MO in these regions and tries to connect the start and the goal configuration through a sequence of intermediate collision-free configurations, one corresponding to each narrow region. Thus the global path is synthesized from the segments connecting successive intermediate configurations. The third and the hardest class of the findpath problems that we have considered requires excursions from the initially estimated paths to derive the final paths. Even humans need more than several seconds to solve these problems because a sequential search of the free space is hard to avoid. The sidetracking algorithm allows MO to sidetrack from the candidate path to get proper changes of the orientation of MO.

We have implemented our algorithm for polyhedral objects. Thus, it can be applied almost directly when an octree representation is given, since cubes are simple polyhedra. We have tested the algorithm on a variety of 2-D and 3-D examples, some of which are shown in Figure 6. Details can be found in (Hwang & Ahuja, 1988 a, b).

### 1.4. Parallel Architectures

To design an efficient architecture requires taking into account the characteristics of the algorithms to be executed by the architecture. A major feature of image data is its planar nature, and an important feature of many image computations is that they are spatially local. Together these two characteristics imply that parallel subtasks may be generated by using the *divide-and-conquer* paradigm to perform computations on subimages, in parallel, and by merging their results.

We have described *multiprocessor pyramids* as one example of an architecture that uses a hierarchical organization of processing elements (PE's) to implement the divide-and-conquer paradigm (Ahuja & Swamy, 1984a,b). The central feature of these architectures is a hierarchy imposed on the image by a recursive square decomposition of the image. The image is overlaid with a sequence of increasingly fine square tessellations that define a recursive embedding of, and thus a hierarchy over, image windows. The hierarchy is described by a complete tree whose root node is associated with the entire image (Figure 7). Each node in the tree represents a square window. Each nonleaf node has four children. Each child node is associated with a quadrant of the parent window. Leaf nodes correspond to pixels or to windows of the smallest size.

The pyramid structure holds the promise of being a natural environment for parallel execution of divide-and-conquer algorithms. It also allows fast message passing between distant nodes. However, we observed that a simple-minded translation of a pyramid model into hardware *fails* to provide a cost-effective, high-performance system. We arrived at this conclusion as a result of a detailed simulation of the execution of a purely bottom-up algorithm on a pyramidal structure wherein each interior node has four children. The algorithm generates the Voronoi diagram for a set of planar points using the divide-and-conquer approach. We have devised a highly flexible multiprocessor system called NETRA for image computations. NETRA is motivated by the pyramid architecture but it attempts to avoid the serious problems associated with direct, physical pyramidal interconnection among processors.

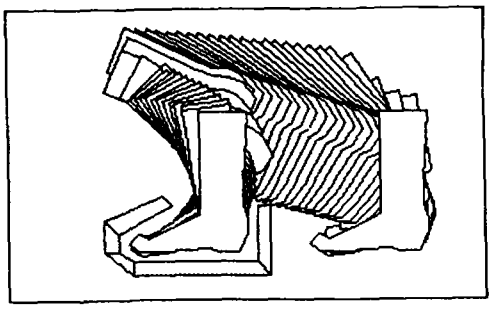
The overall structure of NETRA is recursively defined in a tree-type hierarchy. Leaves of the tree are clusters of processors. The internal nodes are processors, their primary function being task scheduling and load monitoring. In the final level of task schedulers, each task scheduler has a processor cluster as its leaf successor. No degree of the scheduler tree is implied. As a matter of fact, the degree need not even be uniform.

The major feature of NETRA is the fact that it is *algorithm driven*, i.e., its design is determined by general computational characteristics of the algorithms it is intended to execute; it is not designed to perform efficiently on some selected types of image operations. The divide-and-conquer paradigm is a powerful method of parallelizing image understanding computations, and NETRA constitutes a "divide-and-conquer machine."

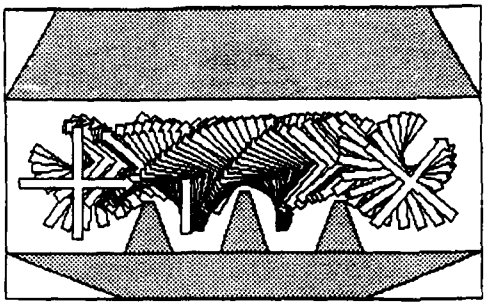
The traditionally used array processors are unable to handle complex mathematical functions and complex data structures that are required for high-level operations in image understanding. Similarly, many traditional, special purpose, pipeline architectures are designed to control data flow to efficiently perform certain low level computations. Therefore, even if such processors, e.g., WARP and CAAP, are available for lower level image processing operations, we must still have some other multiprocessor to handle high-level operations. NETRA can use machines such as WARP and CAAP as slaves to combine their power of efficiently executing low level operations with NETRA's own capability of efficient task parallelization and parallel execution.

NETRA should be as easy to program as any general purpose machine, since the algorithms need only use the high level, divide-and-conquer paradigm. No familiarity with the hardware is necessary. This is another advantage of NETRA over some other special purpose machines which are non-trivial to program.

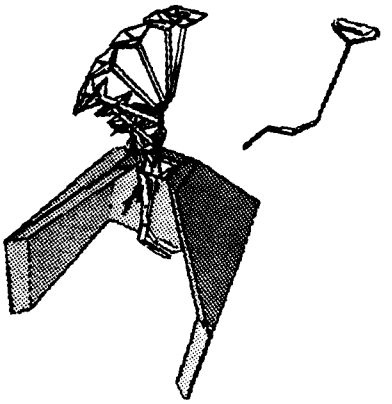
NETRA is expected to overcome the problems mentioned earlier with pyramid and other architectures. Two important factors contribute to this improvement. *First*, the flexibility of configuration allows tasks to be executed in a manner that is most efficient for the particular task. These include dynamically configured pipelines, special-purpose interconnection patterns for sorting, FFTs, etc., and an MIMD mode that is most suitable for problems characterized by a great inhomogeneity of computational requirements across image, e.g., generating the Voronoi diagram of a dot pattern. *Secondly*, the clusters of processing elements do not correspond to fixed regions in images. Tasks are allocated to clusters not on the basis of their spatial locality but in a manner that heuristically distributes the load evenly over the entire system.



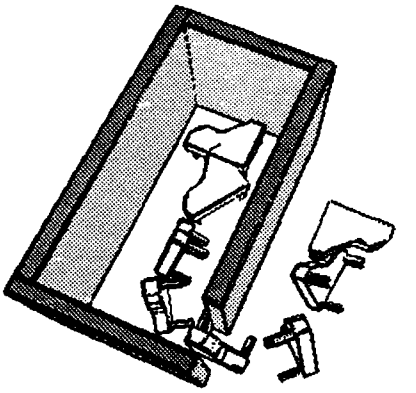
A rigid foot is putting on a shoe.



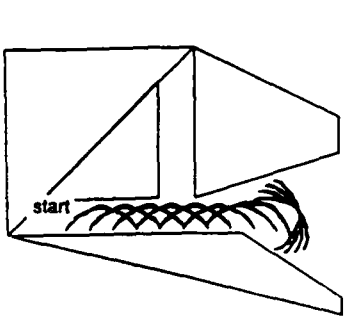
An X shape goes through a series of rotations to pass three narrow bottlenecks.

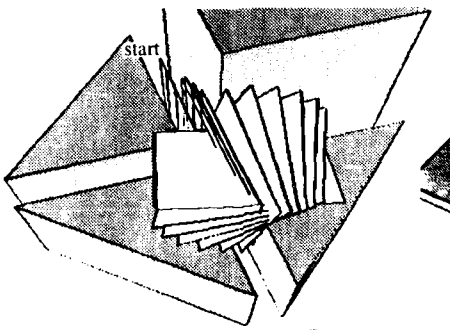


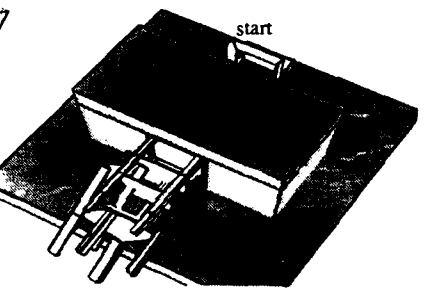
A flower with a crooked root is being pulled out of a vase (the front wall is not shown).

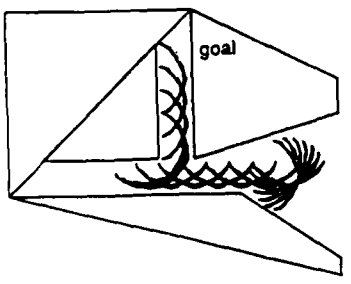


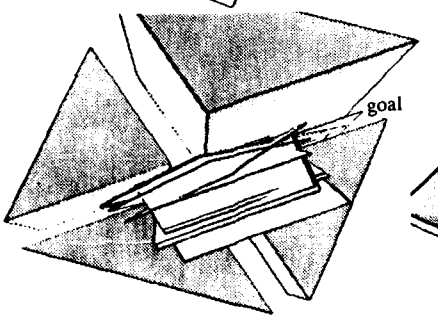
A grand piano is being moved into a room with a narrow doorway.

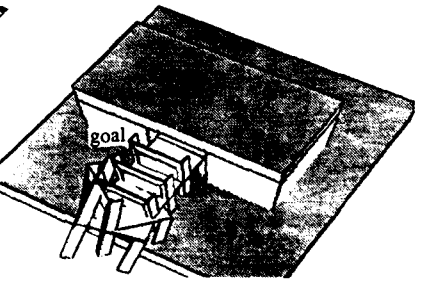


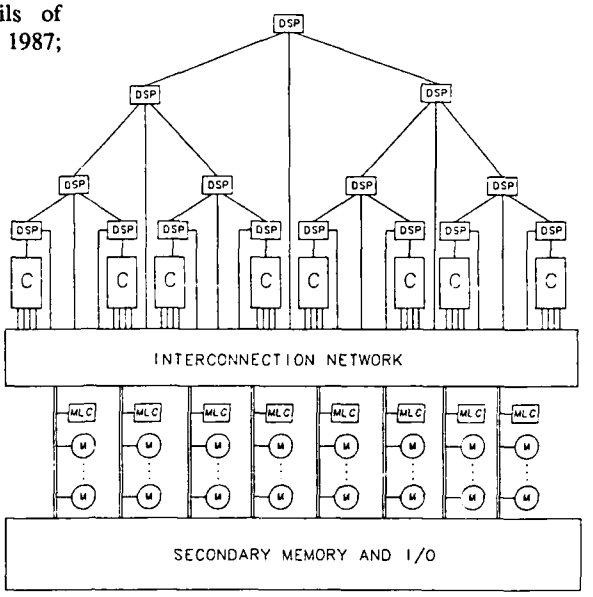

The arc needs to sidetrack from the T-shaped junction to the right first (top), then to the left (bottom) to reach the goal from the start.

A square object sidetracks to the free space above to make a turn at the crossroad.

The problem of moving a chair from one side of a desk to the other. The chair sidetracks twice to change its orientation.

Figure 6. Examples of Path Planning in two and three dimensions.

Since NETRA executes divide-and-conquer algorithms, the utility of these algorithms goes *beyond* NETRA, to any system that implements the divide-and-conquer paradigm. This property is a direct consequence of the fact that NETRA is designed for efficient execution of *all* levels of image understanding operations: it is not designed to perform very efficiently on only certain specialized tasks. Details of NETRA can be found in (Sharma, Ahuja & Patel, 1987; Sharma, Patel & Ahuja, 1985).




C: Processor Cluster
DSP: Distributing and Scheduling Processor
M: Memory Module
MLC: Memory Line Controller

Figure 7. The organization of NETRA.

## 2. SHAPE FROM TEXTURE BY INTEGRATING TEXTURE-ELEMENT EXTRACTION AND SURFACE ESTIMATION•

### 2.1. Introduction

Texture variations provide important cues for recovering the three dimensional structure of the surfaces visible in an image. A uniformly textured surface undergoes two types of projective distortions during the imaging process. Firstly, an increase in the distance from the surface to the viewer causes a uniform compression of increasingly large areas of surface onto a fixed area of image. Secondly, as the surface slants away from the image plane, foreshortening causes an anisotropic compression of the texture. The resulting texture gradients provide information about the relative distances and orientations of the textured surfaces visible in an image.

Projective distortion affects many texture features. Consider an idealized texture showing the perspective view of slanted a planar surface covered with nonoverlapping circular disks of constant size. The disks project as ellipses in the image. The major axis of each ellipse is perpendicular to the tilt[1], whereas the minor axis is parallel with the tilt. The apparent size of the major axes decreases linearly in the direction of tilt, due to increasing distance from the viewer. The apparent size of the minor axes decreases more rapidly: in addition to the distance scaling, the minor axes are reduced by increasing foreshortening. (Foreshortening is inversely proportional to the cosine of the angle between the line of sight and the surface normal.) These changes in the major and minor axes cause an increase of the eccentricity of the ellipses in the tilt direction. The area of the ellipses decreases fastest in the direction of tilt. This is accompanied by an increase in

•This section is part of a paper by Blostein and Ahuja, currently under review at IEEE PAMI.

[1] We express surface orientation in terms of two angles, *slant* and *tilt* (Stevens [1983]). Slant, ranging from 0° to 90°, is the angle between the surface and the image plane. Tilt, ranging from 0° to 360°, is the direction in which the surface normal projects in the image; a tilt of 0° indicates that distance to the viewed surface increases fastest toward the right side of the image.

the density of the ellipses. In this idealized texture, the uniformity in the size, shape and placement of the texture elements (*texels*) leads to pronounced texture gradients.

Natural textures are much less regular than the idealized disk texture mentioned earlier; therefore the texture gradients are not as easily observed (Ahuja, 1987). Natural textures display considerable variability of texel size, shape, coloration and density. Physical texels are typically three-dimensional, in contrast with the two-dimensional disks. This three-dimensionality results in highlights and shadows, and in occlusions between one texel and the next. Also, physical texels have a complex structure. In contrast to a uniform synthetic disk, a physical texel changes in appearance as the resolution is increased: subtexture becomes visible. In an image with fixed resolution, more subtexture is visible for the nearby texels than for the distant texels. Supertexture regions arise when small image texels, corresponding to distant physical texels, blur into larger regions of relatively homogeneous gray level. These factors make it difficult to identify texture elements and extract texture gradients from real images.

This section describes our work on how to exploit textural cues to infer the relative distance and orientation of the textured surfaces depicted in an image. We do not address the problem of texture discrimination; we work with images containing only one type of texture. A primary goal of this research is to demonstrate the feasibility of extracting useful measures of texture gradients from images of natural (as opposed to man-made) textures. The textures present on man-made objects frequently exhibit regularities such as parallel lines, perpendicular lines, equally-sized texture elements, or equally-spaced texture elements. Several existing shape-from-texture algorithms exploit these regularities (Kender [1980], Ikeuchi [1980]); however, most naturally occurring textures are too variable to permit successful application of these methods.

A major challenge in texture analysis is to handle scale consistently. Natural surfaces exhibit a rich hierarchy of textures, with each texture element containing subtexture. All texture measurements are prone to distortion due to the presence of subtexture, since the imaging process captures more subtexture details for close texture samples than for distant ones. The algorithms presented here provide good surface-orientation estimates even in the face of significant sub- and supertexture. For a summary of the work reported here, see Blostein and Ahuja [1987b].

## 2.2. The Inference of Surface Shape

Since texture properties vary across the image in a manner predictable from the physical texture and surface shape, it should be possible to infer surface shape from texture gradients. This section examines the basic requirements of such an inference process. We argue that correct interpretation of texture gradients requires explicit identification of image texels, especially when textures show three-dimensional relief, when texels exhibit significant subtexture, or when it is unknown a priori which texture gradients carry surface-shape information. Texture elements cannot be identified in isolation since texels are defined only by the repetitive nature of the texture as a whole. Therefore, we claim, the identification of texture elements is best done in parallel with the estimation of the shape of the textured surface.

### 2.2.1. Texels

The term *texel*, short for *texture element*, denotes the repetitive unit of which a texture is composed. "Texel" refers to the physical texture element in the real world as well as to the appearance of the texture element in the image. In cases where the distinction must be made, we use the phrases *physical texel* versus *image texel*. Distance and foreshortening changes alter the appearance of the image texel, although the physical texel remains unchanged.

We restrict image texels to be regions of relatively uniform gray level. Under this definition, a physical texel can give rise to several image texels: typically the physical repetitive unit of a texture contains both bright and dark regions. As described below, we treat the bright and dark image texels as separate texture fields. Requiring an image texel to have "relatively uniform" gray-level means that the texel is uniform relative to the gray-level changes that occur at its own scale; however, the texel may contain significant internal variations of gray level. In other words, large texels appear as regions of uniform gray-level only after suitable blurring of the original image.

### 2.2.2. Texture fields

We use the term term *texture field* or *field of texels* to denote a collection of image texels that exhibit one or more consistent texture gradients. Consistency is defined with respect to the texture gradients expected from a particular surface arrangement viewed under perspective. There are several common reasons for several texture fields to occur in a single image. Firstly, many textures are composed of closely associated bright and dark fields which arise from lighting effects. For example, the aerial view of houses in Figure 10(a) contains a field of bright texels composed of the houses and a field of dark texels composed of the shadows cast by the houses. Secondly, associated bright and dark texture fields can arise from different components of the physical structure of the texture elements; see, for example, the sunflowers in Figure 15(a). Thirdly, it is possible for physically separated textured surfaces to be spatially interleaved in an image. This is strikingly illustrated by the birds over water shown in Figure 11(a), where the birds and the water are located in two physically separated planes. Finally, multiple texture fields result from physical surfaces that are covered by several types of texture elements. An aerial view of a residential neighborhood shows one texture field consisting of houses and another texture field consisting of trees.

The concept of texture field is useful for separating portions of physical texels that exhibit differing foreshortening properties. Consider, for example, an aerial view of many flat-roofed houses. The roofs of the houses, which are parallel to the textured plane, are foreshortened increasingly as the angle between the line of sight and the plane decreases, whereas the walls of the houses exhibit the opposite behavior since they are perpendicular to the textured plane. Any analysis of foreshortening in such an image must treat these two texture fields separately. The difference in gray-level

properties of the two fields can help to achieve this separation.

The unknown and often statistical nature of texture regularities makes it impossible to judge a priori whether two texture elements belong to the same texture field. The perception of a texture field is the result of an aggregation phenomenon that requires a consistent texture gradient across the whole field. All the texels belonging to a texture field must be recognized simultaneously.

### 2.2.3. The importance of texture element identification

The extraction of texture elements is an essential step in texture analysis. Texel identification permits correct analysis of textures containing subtexture. Explicit texel identification also permits a unified treatment of the various texture gradients that may be present in an image. Previous researchers' have avoided texel identification because it is quite difficult to do in real images[2]. Instead, indirect methods are used to estimate texel features. We give below several examples of such methods, and indicate why these methods may give erroneous results on many images.

Most previous shape-from-texture algorithms use indirect methods to estimate texel features, by making some assumptions about the nature of texture elements. For example, texel density may be estimated by measuring edge density, under the assumption that all detected edges correspond to the borders of texture elements (Rosenfeld [1975], Aloimonos and Swain [1985], Aloimonos [1986], Kanatani and Chou [1986]). Alternatively, texture elements may be assumed to have uniform edge direction histograms; surface orientation can then be estimated from any deviations from isotropy observed in the distribution of edge directions (Witkin [1981], Davis et al. [1983], Kanatani [1984]). Texture coarseness and directionality may be characterized using Fourier domain measurements (Bajcsy and Lieberman [1976]), making the assumption that only texel characteristics are captured by the Fourier domain measurements. Many man-made textures have parallel, perpendicular and equal-length lines. Assuming that the texel borders can be detected in the image, the surface orientation can be estimated from the observed angles between line segments in the image (Kender [1980], Ikeuchi [1980], Nakatani et al [1980]).

All of these methods may encounter errors when applied to complex natural textures seen under natural lighting conditions[3]. Since texture elements are not identified and explicitly dealt with, it becomes difficult to distinguish between responses due to texture elements and those due to other image features. For example, edge density measurements include contributions from subtexture or supertexture edges, from borders of partially occluded texture elements, and from edges of texels belonging to several texture fields. Similarly, when making an edge-direction histogram it may not be possible to distinguish between edges from texel borders and edges due to other features such as subtexture. Fourier domain features are also sensitive to the presence of subtexture and supertexture. It appears to be necessary to recognize the texture elements before the various measures can be computed as intended.

This problem is illustrated by Figure 8, which shows all of the edges extracted from the texture image of a rockpile (Figure 9(a)). We use an edge operator described by Nevatia and Babu [1980]. Six 5-by-5 edge masks at different orientations are used; the mask giving the highest output at each pixel is recorded. The edges are thinned by suppressing non-maxima perpendicular to the edge directions. The exact details of the edge operator are not important here. We merely wish to illustrate that it would be incorrect to interpret all of the detected edges as boundaries of texture-elements. Additional edges arise due to sub-texture and due to the presence of several texture fields in a single image; these edges are not artifacts of this particular edge detector, since they are clearly present in the original images. Many natural textures have a hierarchical physical structure that causes observed edge density to be nearly constant throughout the image: edges from subtexture and sub-subtexture are observed to whatever detail the camera resolution permits.

In the early stages of this research we experimented with measurements of edge density to detect texture gradients. To eliminate sub texture edges, we suppressed weak edges in the vicinity of strong edges. This was somewhat successful, since the contrast of subtexture is usually less than the contrast of the texture elements themselves. This edge suppression is an indirect attempt to identify texture elements, since the goal is to suppress all edges except those that result from the boundaries of texture elements. We abandoned the edge-based approach in favor of a region-based approach, in which the problem of texel identification is approached more directly, and can thus be solved in a more general way.

Explicit identification of texture elements offers an additional advantage: texture elements provide a unifying framework for examination of the various texture gradients (gradients of apparent texel area, aspect ratio, density etc.) that may be present in an image. A given image may exhibit any combination of texture gradients. The relative accuracy of the gradients, in general, varies from image to image[4]. Since it is not known in advance which texture gradients are useful for estimation of the three-dimensional layout of a given scene, a shape-from-texture system should analyze the variations in different textural properties, and selectively pay attention to the relevant and accurate gradients.

[2] Ohta et al. [1981] use the observed areas of pairs of texels to obtain vanishing points. However, the method has been tested only on synthetic texture images. The problem of extracting texels from natural images is not addressed.

[3] Shape-from-texture algorithms are sometimes tested on images formed by artificial projections derived from images of frontal texture samples (either by digitizing a perspective view of a photograph of the frontal texture, or by using a computer program to simulate such a perspective projection). Such projections introduce simplifications, because physical relief and subtexture are not accurately modeled (texels do not shadow or occlude each other, they foreshorten improperly, no subtexture details appear when regions of the frontal texture sample are expanded to model parts of the surface that are close to the viewer). Therefore, texture algorithms that successfully handle images derived from frontal texture samples cannot necessarily cope with real images of slanted physical textures.

### 2.2.4. Integration of texel identification and surface shape estimation

Texel identification is difficult because texture elements have tremendously varied shapes, sizes and gray-level characteristics. Regions of relatively uniform gray level are appropriate primitives for texel recognition. However, only a subset of the uniform image regions correspond to texture elements. Other regions arise from subtexture within close-range texture elements, from supertexture containing several distant texture elements, from partly occluded texture elements, and from isolated objects that are not part of a texture (for example, the snowy areas and the tree trunk in the rock-pile image of Figure 9(a)). An image region is a texel only if the region has properties consistent with the properties of many other image texels; the region must be one among many regions, all of whom are the projections of physical texels distributed along some 3-D surface. Since a surface hypothesis is needed to decide whether a region is a texel, texel identification must be integrated with surface shape estimation.

We have developed a two-step approach to carry out such integration of texel identification and surface estimation. First, we assume that all homogeneous gray-level regions are candidates for being texels; the first step performs a local gray-level analysis to identify potential texels. Second, we use surface-fitting to identify the true texels from among the candidates, while simultaneously constructing an approximation to the shape of the textured surface. The second step thus enforces perspective viewing constraints to select texels. The next three sections present the algorithm that we have implemented. Section 2.3 describes a region detector for extracting candidate texture elements. The surface-fitting algorithm is described in Section 2.4. Finally, Section 2.5 contains a summary of the implementation, and presents results for a variety of images of textured natural scenes.

## 2.3. Identifying Candidate Texture Elements: Multi-scale Region Detection

Any region that has relatively uniform gray-level is a candidate texel. The uniformity of small regions is measured relative to a small surrounding neighborhood in the image, whereas the uniformity of large regions is measured relative to a proportionally larger neighborhood in the image. The list of candidate texels must include uniform regions of all shapes and sizes. We simplify the problem of extracting regions of arbitrary shapes and sizes by assuming that each region can be represented as a union of overlapping circular disks. Large disks define the rough shape of a region, with overlapping smaller disks capturing finer shape details such as protrusions and concavities. Below we describe a method of extracting all circular image regions of relatively uniform gray level. Sets of overlapping circular regions are then used to form candidate texture elements.

Our region detector is based on the image response to convolution with $\nabla^2 G$ filters over a range of scales. Related work includes Witkin [1983] (a scale-space representation of $\nabla^2 G$ zerocrossings) and Crowley and Parker [1984] (a representation of $\nabla^2 G$ peaks and ridges over a range of scales[5]). We find circular image regions of uniform gray level by convolving the image with $\nabla^2 G$ masks over a range of scales, and comparing the convolution output to that expected for an ideal circular disk of constant gray level.

### 2.3.1. A closed form expression for the $\nabla^2$G response of a disk

The algorithm for uniform-region extraction is based on calculations of the $\nabla^2 G$ and $\frac{\partial}{\partial\sigma}\nabla^2 G$ responses of a disk image. Here we present a brief summary of the region detection algorithm. This region detector is discussed and analyzed in greater detail in Blostein [1987a], Blostein and Ahuja [1987c].

Given a function $I(x,y)$ which specifies the intensity of an image, the $\nabla^2 G$ response of this image at $(x,y)$ is given by the following convolution:

$$\nabla^2 G(x,y) * I(x,y) =$$

$$\int_{-\infty}^{+\infty}\int \frac{2\sigma^2-(u^2+v^2)}{\sigma^4}\, e^{-(u^2+v^2)/2\sigma^2} I(x-u,y-v)\, du\, dv \qquad (1)$$

Mathematical analysis of the response of the $\nabla^2 G$ filter to most images is difficult because the convolution integrals of Equation (1) do not have closed form solutions. However, a closed-form solution can be derived for the center point of a circular disk of constant intensity. We analyze the $\nabla^2 G$ response at the center of an ideal circular disk in the continuous domain; to generate the $\nabla^2 G$ convolution of digitized images, we sample the $\nabla^2 G$ filter values and perform a discrete convolution. The image of a disk of diameter $D$ and contrast $C$ is defined by

$$\text{disk image:} \quad I(x,y) = \begin{cases} C & \text{if } x^2+y^2 \le D^2/4 \\ 0 & \textit{elsewhere} \end{cases} \qquad (2)$$

Using this definition of $I(x,y)$ in Equation (1), and setting $x$ and $y$ to zero, we find (Blostein [1987a], Blostein and Ahuja [1987c]) that at the disk center

[4] This may be illustrated by the following examples. It is common for physical texels to be fairly uniform in size and shape, but for the gaps between the texels to be much less uniform (Figures 11(a), 17(a), 19(a), and 20(a)). In these images, it is more accurate to infer a three-dimensional surface from the size and aspect-ratio gradients than from the density gradient or the gradient of spacings between texels. As a second example, the potential accuracy of the aspect ratio gradient is higher in textures where the physical texels are separated by gaps than in textures where the physical texels overlap and occlude one another (the lilly pads in Figure 20(a) show a much better aspect ratio gradient than do the rocks in Figure 9(a)). Thirdly, for the water hyacinths of Figure 22(a), the random three-dimensional arrangement of the leaves makes the aspect ratio gradient very weak, while the area gradient is still quite significant. Finally, in images with partial occlusions (Figures 14(a) and 15(a)), the perspective gradient (length of the unforeshortened texel dimension) is more accurate than the area gradient: if only part of a texel is occluded, the apparent texel area is decreased, whereas the complete unforeshortened dimension may remain in view.

[5] Crowley and Parker use a difference-of-Gaussian operator, which is a discrete approximation to $\frac{\partial}{\partial\sigma}G$ and hence to $\nabla^2 G$. By the diffusion equation, $\nabla^2 G = (1/\sigma)\frac{\partial}{\partial\sigma}G$.

$$\nabla^2 G \text{ response} = \frac{\pi C D^2}{2\sigma^2} e^{-D^2/8\sigma^2} \quad (3)$$

$$\frac{\partial}{\partial\sigma}\nabla^2 G \text{ response} = \frac{\pi C D^2}{2}\left(\frac{D^2}{4\sigma^5} - \frac{2}{\sigma^3}\right) e^{-D^2/8\sigma^2} \quad (4)$$

Dividing these expressions, we solve for the diameter D and contrast C of the disk:

$$D = 2\sigma\sqrt{\sigma\left(\frac{\partial}{\partial\sigma}\nabla^2 G * I\right)/(\nabla^2 G * I) + 2} \quad (5a)$$

$$C = \frac{2\sigma^2}{\pi D^2} e^{D^2/8\sigma^2}(\nabla^2 G * I) \quad (5b)$$

where the convolutions are evaluated at the center of the disk.

### 2.3.2. Extracting candidate texture-elements in real images

We construct an approximation of image texture elements by first detecting disks of homogeneous gray levels in the image, and then identifying spatially disjoint subsets of disks such that disks in each subset are connected through pairs of overlapping disks. For the first step, we use equations (5) to estimate region diameter (along with region contrast) from the $\nabla^2 G * I$ and $\frac{\partial}{\partial\sigma}\nabla^2 G * I$ values at the center of a region. We choose extrema of the $\nabla^2 G * I$ images as possible disk-center locations. The disks fit to local maxima have positive contrast (regions brighter than the surround), whereas the disks fit to local minima have negative contrast (regions darker than the surround). We use a range of filter sizes. For a given region, local $\nabla^2 G$ extrema occur at the region center for filter sizes that approximately match the diameter of the region ($w \approx D$, $\sigma \approx 2\sqrt{2}D$). To fit disks as accurately as possible, we accept a disk only if the computed diameter $D$ is close to the width of the $\nabla^2 G$ center-lobe: $\sigma \approx 2\sqrt{2}D$. Details of the implementation are covered in Section 2.5.

In the second step we form a union of overlapping disks to construct an approximation of the texture elements. When overlapping disks are grouped together, concavities are formed at the joins between the disks. Some concavities arise at the border between two neighboring texels; at other times the concavities are part of the shape of an individual texel. There is no a priori way to tell whether a set of disks should be split at a concavity or not. For each set of disks, we therefore treat all possible subsets of disks (split and unsplit) as mutually exclusive candidate texels. Some implementation details are given in Section 2.5. For further details see (Blostein and Ahuja, [1987b]).

## 2.4. Surface Estimation and Texel Identification

Our goal in analyzing image texture is to find a spatial layout of homogeneously textured surfaces that could result in the given image texture. We do this by testing many spatial layouts and choosing the one that is the most consistent with a maximal subset of the candidate texels. The surface parameters are determined at the same time that the true texels are chosen from among the candidates.

### 2.4.1. The expected distribution of texel areas for a planar surface

The current implementation is restricted to fitting a single planar surface to the image, based on the observed areas of the candidate texture elements. In order to find a planar fit to the candidate texels, we need to know the distribution of texel areas that occurs in an image of an idealized textured plane. To derive this relationship, we assume a planar textured surface covered with identical texels, where the texels show no three-dimensional relief. Natural textures are typically more complicated: they are composed of highly variable texels that show three-dimensional relief. Nevertheless, our experiments show that the equations derived from consideration of idealized textures are useful for analyzing a variety of natural textures as well (Section 2.5).

In Blostein [1987a], Blostein and Ahuja [1987b] we derive expressions for spatial variation in the dimensions and area of texels in idealized, planar textures. We express the texel parameters as a function of image location in the direction of greatest depth increase. For example, the area $A_i$ of a texel anywhere in the image may be expressed as

$$A_i = A_c(1 - \tan\theta \tan S)^3 \quad (6)$$

where $A_c$ is the texel area measured at the image center, $S$ is the slant, and $\theta$ is an angle that depends on the tilt-direction component of the vector from the image center to the current image location. To measure $\theta$ given an image location, a tilt, and the field-of-view of the camera lens, project the view-ray onto the plane through the focal point that is parallel to the tilt and perpendicular to the image plane. Then $\theta$ is the angle between this projected view ray and the optic axis.

### 2.4.2. Fitting a planar surface to the candidate texture elements

Having extracted candidate texels from an image of a textured surface, we find the orientation of the textured plane that best agrees with the observed areas of the candidate texels. A planar surface is characterized by the triple $(A_c, S, T)$, where $A_c$ is the texel area expected in the image center, $S$ is the slant, and $T$ is the tilt. In order to find the best planar fit for the image texture, we discretize the possible values of $A_c$, $S$ and $T$, and evaluate each possible planar fit. For each choice of $(A_c, S, T)$, Equation (6) gives the expected texel area at each image location. These expected areas are compared to the region areas actually occurring in the image, and a fit-rating is computed for the plane. The plane that receives the highest fit-rating is selected as the estimate of the textured surface. The candidate texels that support the best planar fit are interpreted as true image texture elements. The rating of a planar fit is computed as

$$\text{fit-rating} = \sum_{\text{all regions}} (\text{region area})\ |\text{region contrast}|\ e^{-(\text{region-fit})^2/4} \quad (7)$$

$$\text{where region-fit} = \frac{\max(\text{expected area, actual area})}{\min(\text{expected area, actual area})}$$

The region-fit is 2.0 for a candidate texel that is either half as big or twice as big as the size predicted by the planar fit. We

begin with a coarse fit, in which the $(A_c, S, T)$ space is searched at sparse locations. To refine the planar fit, a more detailed search of the $(A_c, S, T)$ space is done in the neighborhood of the best plane from the coarse fit. The fit-rating values change smoothly as a function of $A_c$, slant and tilt. Section 2.5 gives some implementation details. For further details, see (Blostein and Ahuja, [1987b]).

### 2.5. Application of the Algorithm to Real Images

Parts (a) of Figures 9 to 25 show the images of natural textures we have used in our experiments. A few of the images are photographs of outdoor scenes taken by one of the authors in Urbana, Illinois. The rest are illustrations in books (see Blostein [1987a] for references), which we have rephotographed. All of these images are digitized off of the photographic negatives using a drum scanner. The images are 512 by 512 pixels; the image sizes in the figures vary because image borders have been trimmed. All of the images are processed the same way; the method has no parameters that need to be tuned to particular images. Before discussing the results on these images, we present some details of the implementation.

#### 2.5.1. Summary of the implementation

Here we list the processing steps used on each of the images shown in Figures 9 to 25. The processing of an image $I$ is divided into three main phases: fit disks to the uniform image regions, construct potential texture elements from the disks, and fit a planar surface to the candidate texels.

**Fit disks to the uniform image regions**

(1) Compute the convolutions $\nabla^2G*I$ and $\frac{\partial}{\partial\sigma}\nabla^2G*I$ for the following six $\sigma$ values: $\sqrt{2}$, $2\sqrt{2}$, $3\sqrt{2}$, $4\sqrt{2}$, $5\sqrt{2}$ and $6\sqrt{2}$. (The center lobes of the six $\nabla^2G$ filters have diameters $w$ of 4, 8, 12, 16, 20 and 24 pixels respectively.) To compute $\nabla^2G*I$ for a particular $\sigma$ value, the image is convolved with a mask whose coefficients are taken from

$$\frac{2\sigma^2-r^2}{\sigma^4}\, e^{-r^2/2\sigma^2}$$

To compute $\frac{\partial}{\partial\sigma}\nabla^2G*I$ for a particular $\sigma$ value, the image is convolved with a mask whose coefficients are taken from

$$\frac{6r^2\sigma^2-r^4-4\sigma^4}{\sigma^7}\, e^{-r^2/2\sigma^2}$$

(2) Mark the locations where disks will be fit. To analyze the positive-contrast regions of the original image, mark all local maxima in the $\nabla^2G*I$ images. To analyze the negative-contrast regions of the original image, mark all local minima in the $\nabla^2G*I$ images.

Local maxima and minima are computed relative to a 3x3 neighborhood.

(3) At each marked location, use the measured $\nabla^2G*I$ and $\frac{\partial}{\partial\sigma}\nabla^2G*I$ values to compute a disk diameter and disk contrast:

$$D = 2\sigma\sqrt{\sigma(\tfrac{\partial}{\partial\sigma}\nabla^2G*I)/(\nabla^2G*I) + 2}$$

$$C = \frac{2\sigma^2}{\pi D^2}\, e^{D^2/8\sigma^2}\, (\nabla^2G*I)$$

Retain only the disks where $w-2 \le D \le w+2$ ($w$ is the width in pixels of the center lobe of the $\nabla^2G$ filter). Form a single set of disks by taking the union of the disks detected at the various filter sizes.

**Construct potential texture elements from the disks**

To form the list of potential texture elements, extract all subsets of disks that are spatially connected and contain no concavities greater than 90°. If a concavity is in the range 50° to 90°, use the disks to form three potential texture elements[6]: one large region consisting of all the disks, and two smaller regions resulting from splitting the large region at the concavity[7]. Mark mutual exclusion between potential texture elements that share a disk: at most one of them can contribute support to a planar fit and be chosen as a true texture element.

**Fit a planar surface to the candidate texels**

$A_c$ is the texel area expected in the image center, $S$ is the slant, and $T$ is the tilt of a hypothesized planar fit. For a coarse fit, choose $A_c$ (in units of pixels) from the set {10, 20, 40, 80, 160, 320, 640}, choose $S$ from {0°, 5°, 10°, ..., 70°, 75°, 80°}, and choose $T$ from {0°, 20°, 40°, ..., 300°, 320°, 340°}. To perform a fine fit in the neighborhood of the best plane from the coarse fit, change $S$ in increments of 2.5°, $T$ in increments of 5°, and $A_c$ in increments of less than 25%. (The $A_c$ values increase exponentially because area-discrepancies are measured as a ratio of expected to actual areas.)

The expected texel area for a particular choice of $(A_c, S, T)$ is computed as $A_i = A_c(1 - \tan\theta \tan S)^3$. (See Section 4.1. for a definition of the angle $\theta$.) Evaluate a planar fit by adding contributions from each potential texel:[8]

fit-rating =

$$\sum_{\text{all regions}} (\text{region area})\, |\text{region contrast}|\, e^{-(\text{region-fit})^2/4}$$

[6] The particular values 50° and 90° are not critical; we have found that the range 50° to 90° is large enough to capture all regions of interest and yet small enough to prevent a combinatorial explosion in the number of potential texture elements generated.

[7] Region splitting is implemented as follows. We begin with a set $P$ of overlapping disks, which together cover an image region $R$. The largest concavity in $R$ is found by computing the angles formed by every pair of neighboring disks on the border of $R$. Suppose that $X$ and $Y$ are two neighboring disks on the border of $R$, and that they form a concavity that causes a split into smaller, more convex regions. The concavity is split by (1) removing $X$ from $P$ and repeating the above process, and then (2) removing $Y$ from $P$ and repeating the above process.

[8] If potential texture-elements are mutually exclusive (they share a disk), only the potential texel that has the smallest region-fit is included in the fit-rating sum.

where

$$\text{region-fit} = \frac{\max(\text{expected area, actual area})}{\min(\text{expected area, actual area})}.$$

Select the plane that receives the highest fit-rating as the best estimate of the textured surface. Texture elements are those regions that have an area close to the area expected by the best planar fit.

### 2.5.2. Discussion of the results

Figures 9 to 25 illustrate the results we have obtained for 17 images of natural textures. The variety of textures for which the results are presented should help in judging the strengthes, weaknesses and generality of the algorithm and its current implementation. The results obtained for each image are illustrated for the positive-contrast image regions.

The original image is shown in part (a) (top of the column) of Figures 9 to 25. The parameters of the best planar fit are illustrated by the synthetic texture image in part (c) (bottom of the column) of the figures. The detected texels are shown in part (b) (middle of the column): these are all candidate texels having area within a factor of two of the area expected by the best planar fit.

The shape of the fit-rating peak (not displayed here) is related to the properties of the image texture. A sharp fit-rating peak indicates that the texels have small size variance. This is illustrated by the aerial view of houses (Figure 10) and by the field of sunflowers (Figure 15). If the texel sizes have larger variance, as for the clouds (Figure 13) and the rock pile (Figure 9), then the peak is much broader. (In the rock-pile image, the non-planarity of the original textured surface also contributes to the broadness of the fit-rating peak.) The texels shown in part (b) of the figures are those candidate texels having area within a factor of two of the area expected by the planar fit. Using this same factor of two for all images causes incomplete extraction of texels in images where texel size is highly variable. More complete texel extraction can be achieved by adjusting the criteria for choosing texels from the set of candidate texels: the criteria should vary as a function of the broadness of the fit-rating peak in $(A_c, S, T)$ space.

The accuracy of the results may be illustrated in two ways. Firstly, the reader can compare his perception of the textured surfaces (part (a) of Figures 9 to 25) with the planar surface fitted by the program (part (c) of Figures 9 to 25). Agreement with human perception is quite good for many of the images. Secondly, since the processing of the positive-contrast and negative-contrast regions is performed totally independently, the agreement between the slants and tilts obtained by the two analyses strengthens the confidence in the results. (Note that the $A_c$ parameters are not expected to be similar for the positive-contrast and negative-contrast regions -- the positive-contrast and negative-contrast regions may be of very different sizes.) However, the two analyses may not always lead to the same estimates of slant and tilt, because a texture may not be homogeneous in both texel size and texel separation. Thus, an agreement among multiple analyses (such as the two discussed here) must not be required. A method of selecting and integrating the pertinent analyses in a given case must be devised. Such inferencing from gradients of multiple texture properties has not been addressed in the work reported in this paper.

Table 1 summarizes the planar fits obtained for all images. These fits use slants that are multiples of 2.5° and tilts that are multiples of 5°. The slant and tilt values computed from the positive-contrast and negative-contrast regions are often within 10° of each other. Seven of the 17 images have differences less than 10°; nine of the images have differences less than 15°. For reference, a 30° difference in tilt is equal to the angular distance between adjacent numbers on a clock face. A 30° difference in slant, on the other hand, is a more serious error. In many of those images that have a large discrepancy between the two planar fits, attributes of the

**TABLE 1**

| Description | Fit to positive-contrast regions | | | Fit to negative-contrast regions | | | Difference | |
|---|---|---|---|---|---|---|---|---|
| | $A_c$ | slant | tilt | $A_c$ | slant | tilt | slant | tilt |
| A rock pile | 40 | 62.5° | 65° | 40 | 60° | 75° | 2.5° | 10° |
| Aerial view of houses | 35 | 62.5° | 95° | 60 | 67.5° | 110° | 5° | 15° |
| Birds flying over water | 35 | 45° | 80° | 40 | 57.5° | 100° | 12.5° | 20° |
| Prayer at a mosque | 160 | 27.5° | 50° | 120 | 42.5° | 100° | 15° | 50° |
| Fleecy clouds | 100 | 55° | 275° | 160 | 55° | 280° | 0° | 5° |
| 3-D movie audience | 280 | 45° | 105° | 320 | 7.5° | 330° | large | |
| Sunflowers | 160 | 70° | 95° | 200 | 70° | 90° | 0° | 5° |
| A tree trunk | 70 | 65° | 345° | 80 | 42.5° | 0° | 25.5° | 15° |
| Bathers on the Ganges | 100 | 45° | 80° | 80 | 65° | 85° | 20° | 5° |
| A plowed field | 80 | 42.5° | 40° | 100 | 65° | 80° | 22.5° | 40° |
| A field of flowers | 50 | 70° | 90° | 140 | 52.5° | 20° | large | |
| Water lillies | 120 | 75° | 90° | 160 | 52.5° | 70° | 22.5° | 20° |
| Ripples | 50 | 52.5° | 105° | 120 | 62.5° | 105° | 10° | 0° |
| Water Hyacinths | 100 | 37.5° | 80° | 100 | 40° | 80° | 2.5° | 0° |
| The Toulumne River | 25 | 57.5° | 85° | 40 | 65° | 95° | 7.5° | 10° |
| Sand | 240 | 40° | 80° | 200 | 55° | 80° | 15° | 0° |
| Fallen leaves | 40 | 60° | 90° | 50 | 62.5° | 95° | 2.5° | 5° |

original texture lead us to expect the fits to differ in accuracy. We have identified four reasons for the observed discrepancies. In the field of flowers (Figure 19) and the water lillies (Figure 20), the spaces between the texels are less regular than are the areas of the texels; therefore the fit to the negative-contrast regions is not as accurate as the fit to the positive-contrast regions. A second reason the background regions produce inaccurate results is because the properties of the physical texels are more important than the properties of background regions. In images where the physical texels are separated by gaps, the inter-texel spacing carries more information than does the shape or area of the background regions. Thus, the results for the negative-contrast regions of the movie image and the lilly pad image are inaccurate because the area of the background regions poorly reflects the inter-texel spacing. A third reason for discrepancies between the two slant and tilt estimates is a large variability in texel area (as occurs in Figure 12, the image of prayer at a mosque). This causes a broad peak in the planar fit space (part (c) of Figure 12); hence the exact peak location is not as accurate for these images as for others. A fourth reason for inaccurate results is that the current extraction of uniform regions fragments non-compact regions in an arbitrary way, increasing the variabilities of the measured areas. This effect can be seen in the background of the movie image.

For nearly all of the images, at least one of the two analyses produces results that are in good agreement with human perception. Criteria are needed for automatically assessing the accuracy of the results obtained by the two analyses. Our ongoing work may produce a method for estimating the surface orientation jointly from different texture gradients, including the area gradients of positive-contrast regions and negative-contrast regions analyzed in this paper.

### 2.6. Summary

We have presented a general discussion of the problem of recovering scene-layout information from the texture cues present in an image. We argue that extraction of texels is useful and perhaps even necessary for correct interpretation of texture gradients in the face of subtexture and three-dimensional relief. In order to separate texture elements from other regions (such as subtexture) it is necessary to perform texel identification and surface fitting simultaneously.

We have presented an implementation that is based on these ideas; the implementation is restricted to the detection of gradients of texel area. A multi-scale region detector is developed from the response of an ideal disk to convolution with a Laplacian-of-Gaussian ($\nabla^2 G$) over a range of scales. The output of the region detector is used to form a list of candidate texels. These candidate texels then provide the evidence needed to choose a good planar fit to the image texture; at the same time, the best planar fit determines which of the candidate texels are true texels. Results are shown for a variety of natural textures.

A goal of our ongoing research is to estimate surface orientation from an integrated analysis of all relevant texture gradients, including area gradients, aspect-ratio gradients and density gradients. The implementation reported in this paper is a start in this direction.

## REFERENCES

### Section 1 - Self References


N. Ahuja [1987].
"Texture Analysis", S. Shapiro (ed.), *Encyclopedia of Artificial Intelligence*, Wiley, 1987, 1101-1115.

N. Ahuja and M. Tuceryan [1987].
"Dot Pattern Analysis", S. Shapiro (ed.), *Encyclopedia of Artificial Intelligence*, Wiley, 1987, 253-256.

N. Ahuja [1982].
"Dot Pattern Processing Using Voronoi Neighborhoods", *IEEE Trans. Pattern Analysis and Machine Intelligence*, May 1982, 336-343.

N. Ahuja and C. Nash [1984].
"Octree Representations of Moving Objects", *Computer Vision, Graphics and Image Processing*, May 1984, 207-216.

N. Ahuja and S. Swamy [1984a].
"Multiprocessor Pyramid Architectures for Bottom-Up Image Analysis". A. Rosenfeld (ed.), *Multiresolution Image Processing*, Volume 1, Springer-Verlag, 1984, 38-59.

N. Ahuja and S. Swamy [1984b].
"Multiprocessor Pyramid Architectures for Bottom-Up Analysis", *IEEE Trans. Pattern Analysis and Machine Intelligence*, July 1984, 463-475.

N. Ahuja and J. Veenstra [1988].
"Generating Octrees from Object Silhouettes in Orthographic Views", *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1988, to appear.

K. S. Arun, T. S. Huang, and S. D. Blostein [1987].
"Least-squares fitting of two 3-D point sets", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, No. 5, pp. 698-700, 1987.

S. D. Blostein and T. S. Huang [1987].
"Error analysis in stereo Determination of 3-D point positions", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, No. 6, pp. 752-765, 1987.

H.H. Chen and T.S. Huang [1987].
"Multiple object motion estimation by matching 3-D line segments", *Proc. IEEE Workshop on Computer Vision*, Nov 30 -Dec 2, 1987, Miami.

J. Q. Fang and T. S. Huang [1984].
"Some experiments on estimating The 3-D motion parameters of a rigid body from two consecutive image frames", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 547-554, 1984.

D. Goldgof, T.S. Huang, and H. Lee,
"Motion estimation of nonrigid objects", *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, June 6-9, 1988, Ann Arbor., to appear.

W. Gu and T. S. Huang [1985].
"Connected edge extraction from a perspective view of a polyhedron", *IFEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, July 1985, pp. 422-430.

W. K. Gu, J. Y. Yang, and T. S. Huang [1987].
"Matching perspective views of a polyhedron using circuits", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, vol. PAMI-9, no. 3, May 1987, pp. 390-400.

W. Hoff and N. Ahuja [1985].
"Surfaces from Stereo", *Proc. DARPA Image Understanding Workshop*, Miami, December 9-10, 1985, 98-106.

W. Hoff and N. Ahuja [1987a].
"Extracting Surfaces from Stereo Images: An Integrated Approach", *University of Illinois Coordinated Science Laboratory Report* UILU-ENG-87-2204.

W. Hoff and N. Ahuja [1987b].
"Extracting Surfaces from Stereo Images: An Integrated Approach", *First International Conference on Computer Vision*, London, England, June 8-11, 1987, 284-294.

Y. Hwang and N. Ahuja [1988a].
"Path Planning Using a Potential Field Representation", *Proc. IEEE International Conference on Robotics and Automation*, Philadelphia, April 25-29, 1988, to appear.

Y. Hwang and N. Ahuja [1988b].
A Potential Field Approach to Path Planning in Two- and Three-Dimensions, *University of Illinois Coordinated Science Laboratory Report, 1988, to appear.*

Y. Liu and T. S. Huang [1986].
"Estimation of rigid body motion using straight line correspondences", further results, in Proc. *Inter. Conf. Pattern Recognition*, Paris, France, Oct 27-31, 1986, pp. 306-307.

W. Osse and N. Ahuja [1984].
"Efficient Octree Representation of Moving Objects", *Proc. 7th Int. Conf. on Pattern Recognition*, Montreal, Canada, July 30 - Aug 2, 1984, 821-823.

M. Sharma, N. Ahuja and J. Patel [1987].
"NETRA: An Architecture for a Large Scale Multiprocessor Vision System, L. Uhr (ed.), *Parallel Computer Vision*, Academic Press, 1987, 87-105.

M. Sharma, J. H. Patel and N. Ahuja [1985].
"NETRA: A Multiprocessor Computer Architecture for Image Understanding", *IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, Miami, November 18-20, 1985, 92-98.

S. Srivastava and N. Ahuja [1987].
"An Algorithm for Generating Octrees from Object Silhouettes in Perspective Views", *Proc. IEEE Workshop on Computer Vision*, Miami Beach, November 30 -December 2, 1987, 363-365.

R. Y. Tsai and T. S. Huang [1984].
"Uniqueness and estimation of 3-D motion parameters of rigid bodies with curved surfaces", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 13-27, 1984.

R. Y. Tsai and T. S. Huang [1981].
"Estimating 3-D motion parameters of a rigid planar patch", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 1147-1152, 1981.

R. Y. Tsai, T. S. Huang, and W. L. Zhu [1982].
"Estimating 3-D motion parameters of a rigid planar patch, II: singular value decomposition", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 525-534, 1982.

N. Tuceryan and N. Ahuja [1987].
"Extracting Perceptual Structure in Dot Patterns: An Integrated Approach", *University of Illinois Coordinated Science Laboratory Report* UILU-ENG-87-2206.

M. Tuceryan and N. Ahuja [1983].
"Perceptual Segmentation of Nonhomogeneous Dot Patterns", *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Washington D.C., June 19-23, 1983, 47-52.

J. Veenstra and N. Ahuja [1988].
"Line Drawings of Octree-Represented Objects", *ACM Transactions on Graphics*, 1988.

J. Veenstra and N. Ahuja [1986].
"Efficient Octree Generation from Silhouettes", *IEEE Conf. on Computer Vision and Pattern Recognition*, Miami, June 22-26, 1986, 537-542.

J. Weng, N. Ahuja and T. S. Huang [1988a].
"Closed-form solution + maximum likelihood: a robust approach to motion and structure estimation", in Proc. *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, 1988.

J. Weng, N. Ahuja, and T. S. Huang [1988b].
"Motion and structure from point correspondences: a robust algorithm for planar case with error estimation", submitted to *Inter. Conf. Pattern Recognition*, 1988.

J. Weng, T. S. Huang, and N. Ahuja [1987a].
"3-D motion estimation, understanding and prediction from noisy image sequences", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, No. 3, pp. 370-389, 1987.

J. Weng, T. S. Huang, and N. Ahuja [1987b].
"Error analysis of motion parameters estimation from image sequences", in Proc. *Inter. Conf. Computer Vision*, London, England, June, 1987.

J. Weng, T. S. Huang and N. Ahuja [1987c].
"A two-step approach to optimal motion and structure estimation", in Proc. *IEEE Workshop on Computer Vision*, Miami Beach, Florida, Nov. 30 - Dec. 2, 1987, pp 355-357.

J. Weng, T.S. Huang, and N. Ahuja [1988a].
"Determining motion/structure from line correspondences: a robust algorithm and uniqueness theorems", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Ann Arbor.

J. Weng, T. S. Huang and N. Ahuja [1988b].
"Stability of motion and system: error versus motion and system parameters", submitted to *Inter. Conf. Pattern Recognition*, 1988.

J. Weng and N. Ahuja [1987].
"Octrees of Objects in Arbitrary Motion: Representation and Efficiency", *Computer Vision, Graphics and Image Processing*, August 1987, 167-185.

B. L. Yen and T. S. Huang [1983].
"Determining 3-D motion and structure of a rigid body using straight line correspondences", *Image Sequence Processing and Dynamic Scene Analysis*, Heidelberg, Germany: Springer-Verlag, 1983.

**Section 2 - Shape from Texture References:**

Aloimonos, J. and M. Swain [1985].
"Shape from Texture", *Proc 9th International Joint Conference on AI*, 926-931, 1985.

Aloimonos, J. [1986].
"Detection of Surface Orientation from Texture I: The Case of Planes", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 584-593, 1986.

Bajcsy, R. and L. Lieberman [1976].
"Texture Gradient as a Depth Cue", *Computer Graphics and Image Processing*, vol 5, 52-67, 1976.

Blostein, D. [1987a].
*Recovering the Orientation of Textured Surfaces in Natural Scenes*, Ph. D. Thesis, University of Illinois, Coordinated Science Laboratory Report UILU-ENG-87-2219, April 1987.

Blostein, D. and N. Ahuja [1987b].
"Representation and Three-Dimensional Interpretation of Image Texture: An Integrated Approach", *IEEE First International Conference on Computer Vision*, 444-449, June 1987.

Blostein, D. and N. Ahuja [1987c].
"A Multi-scale Region Detector", submitted to *Computer Vision, Graphics, and Image Processing*

Crowley, J. and A. Parker [1984].
"A Representation for Shape Based on Peaks and Ridges in the Difference of Low Pass Transform", *IEEE PAMI*, vol 6, no 2, 156-170, March 1984.

Davis, L., L. Janos, and S. Dunn [1983].
"Efficient Recovery of Shape from Texture", *IEEE PAMI*, vol 5, no 5, 485-492, Sept 1983.

Ikeuchi, K. [1980].
"Shape from Regular Patterns (An Example of Constraint Propagation in Vision)", MIT A.I. Memo 567, March 1980.

Kanatani, K. [1984].
"Detection of Surface Orientation and Motion from Texture by a Stereological Technique", *Artificial Intelligence*, 23, 213-237, 1984.

Kanatani, K. and T. Chou [1986].
"Shape from Texture: General Principle", *Proc. IEEE Conf. Computer Vision and Pattern Recognition 86*, Miami, 578-583, June 1986.

Kender, J. [1980].
*Shape from Texture*, Ph. D. Thesis, Carnegie-Mellon University, CMU-CS-81-102, November 1980.

Nakatani, H., S. Kimura, O. Saito and T. Kitahashi [1980].
"Extraction of Vanishing Point and its Application to Scene Analysis Based on Image Sequence", *Proceedings of the International Conference on Pattern Recognition*, 370-372, 1980.

Nevatia, R. and K. R. Babu [1980].
"Linear Feature Extraction and Description", *Computer Graphics and Image Processing*, 13, 257-269, 1980.

Ohta, Y., K. Maenobu and T. Sakai [1981].
"Obtaining Surface Orientation from Texels under Perspective Projection", *Proceedings of the International Joint Conference on Artificial Intelligence*, 746-751, 1981.

Rosenfeld, A. [1975].
"A Note on Automatic Detection of Texture Gradients", *IEEE Transactions on Computers*, vol C-24, 988-991, October 1975.

Stevens, K.A. [1983].
"Slant-Tilt: The Visual Encoding of Surface Orientation", *Biological Cybernetics*, vol 46, 183-195, 1983.

Witkin, A.P. [1981].
"Recovering Surface Shape and Orientation from Texture", *Artificial Intelligence*, vol 17, 17-45, 1981.

Witkin, A. P. [1983].
"Scale Space Filtering", *Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, 1019-1022, August 1983.

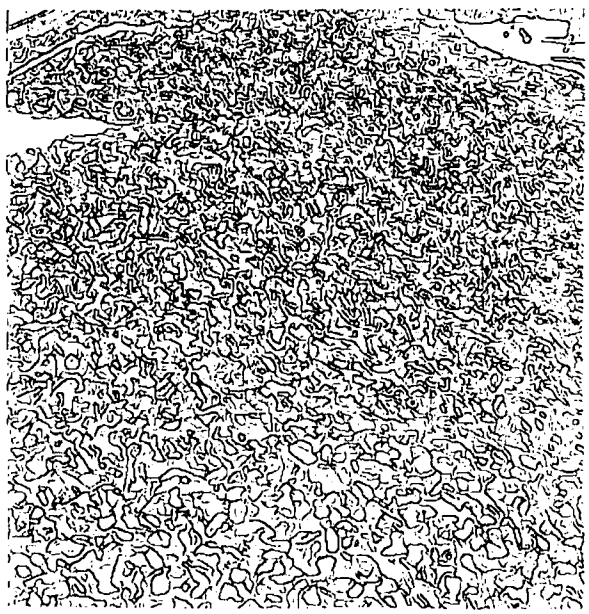


Figure 8. Edges extracted from the rockpile texture. Only a subset of the detected edges are texel boundaries. If edge density is to be effective in capturing the texture gradient, all edges that do not correspond to texel boundaries must be removed. Such edge removal cannot be accomplished without, in effect, performing texel identification.

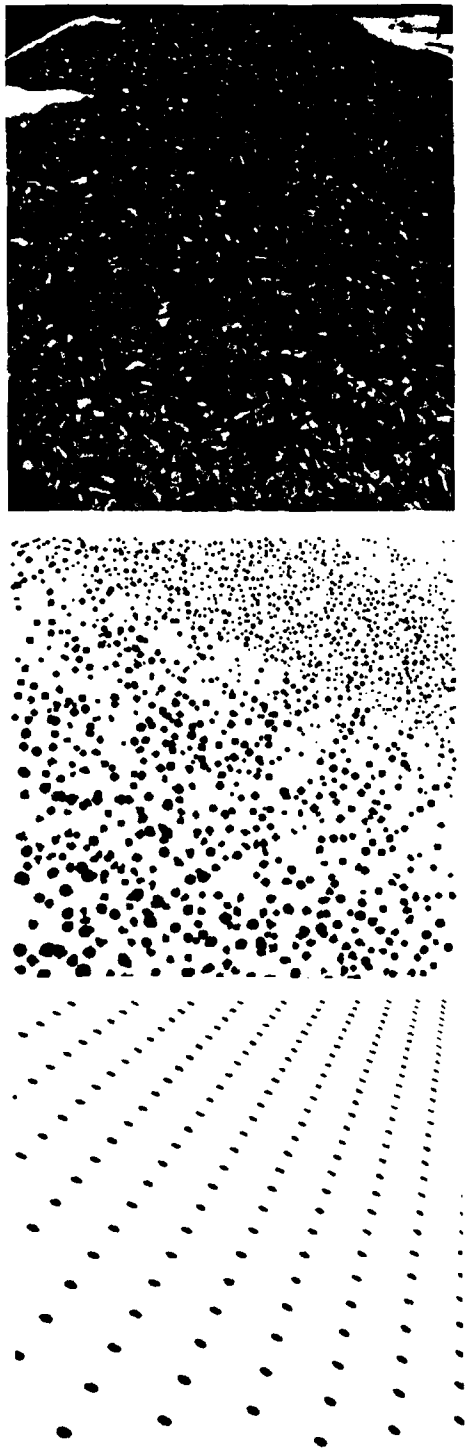


Figure 9. A rock pile.

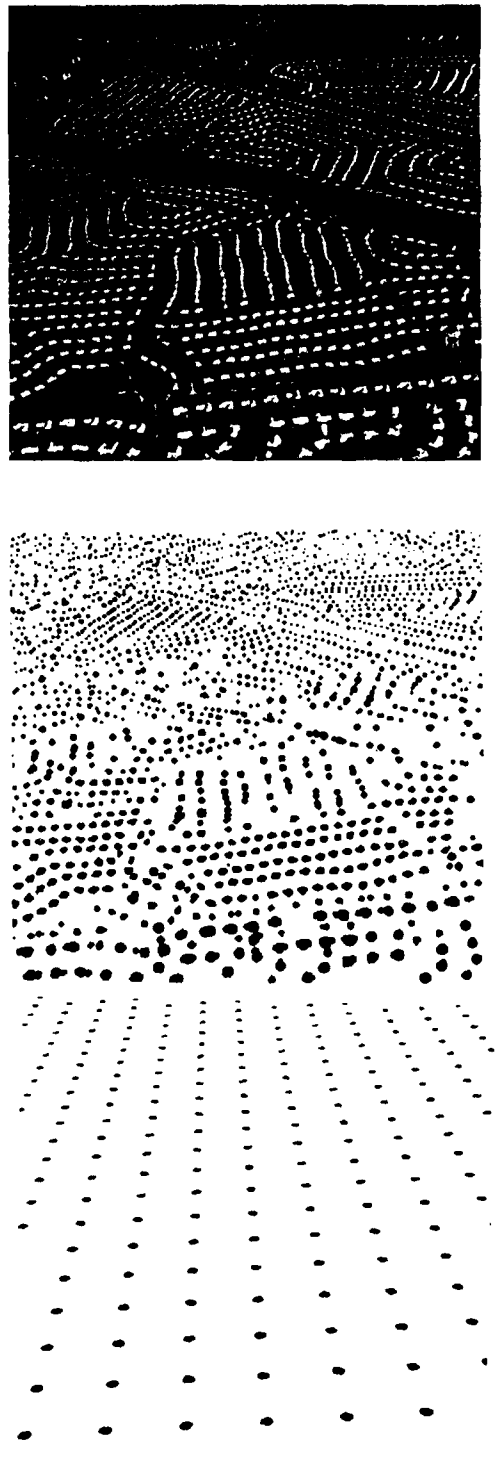


Figure 10. Aerial view of Littown, Pennsylvania.

The figures on this and the following pages (Figures 9 - 25) illustrate the results of integration of texel identification and surface estimation on 17 different textures. The results for each texture are shown as three figures arranged in a column: the top figure shows the original texture; the middle figure shows the detected texels corresponding to the bright (positive contrast) regions; and the bottom figure shows a synthetic image to illustrate the estimated slant and tilt values for the texture.

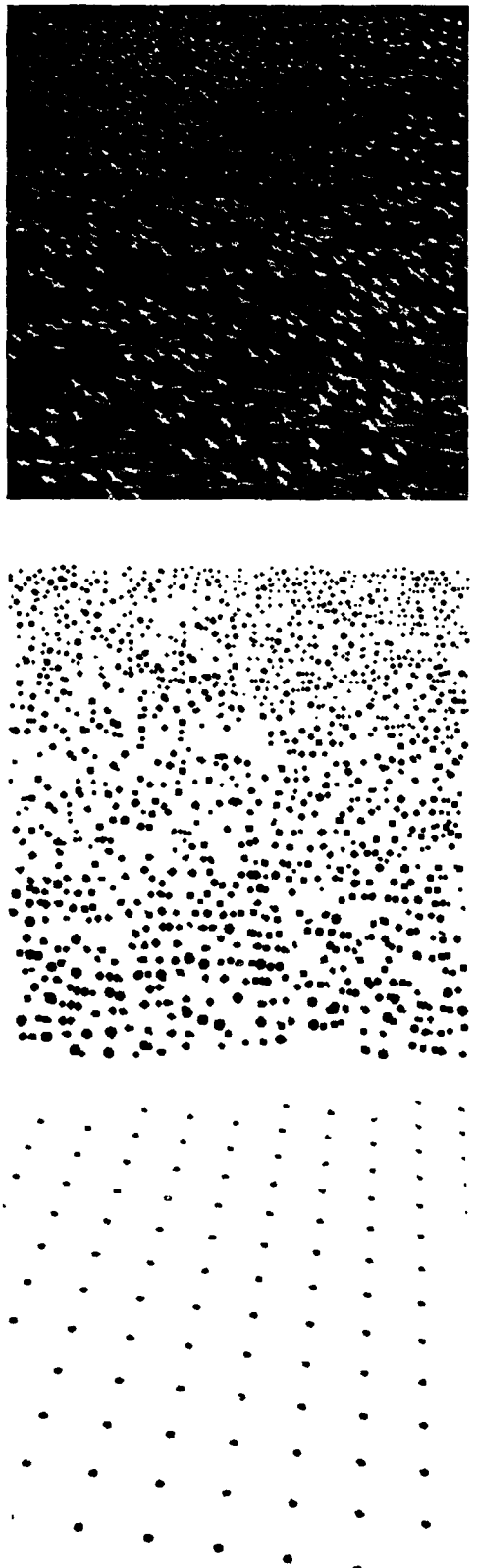


Figure 11. Snow geese over Back Bay, Virginia.

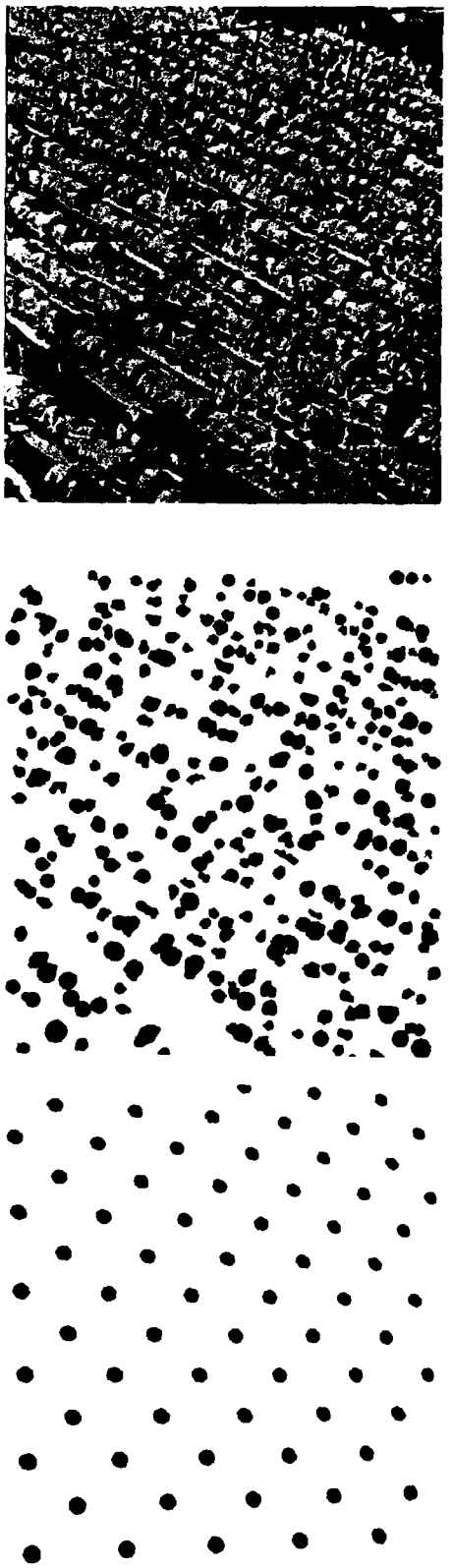
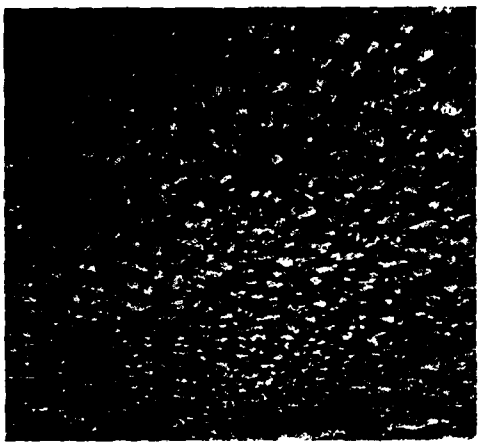
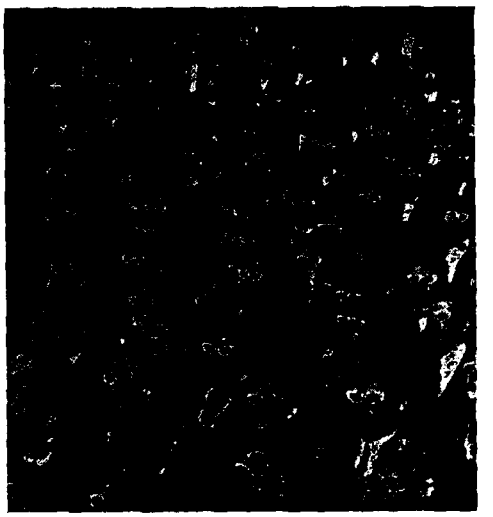
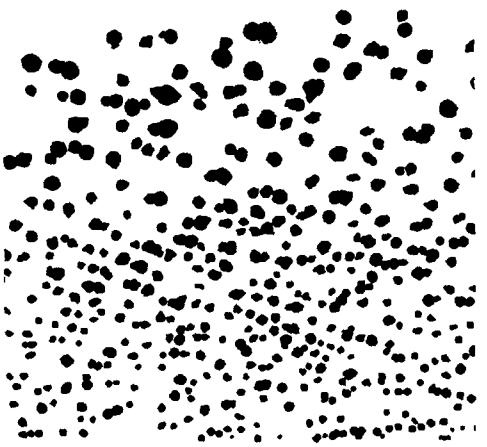
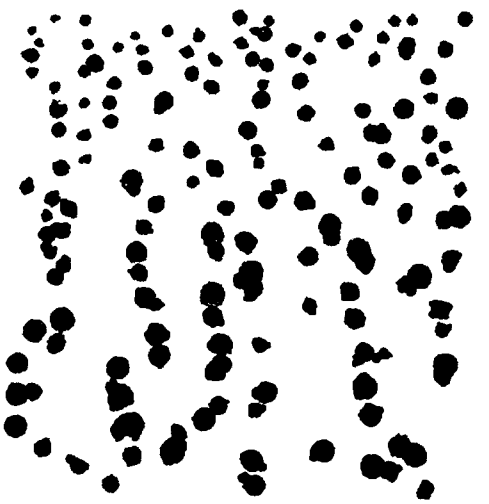


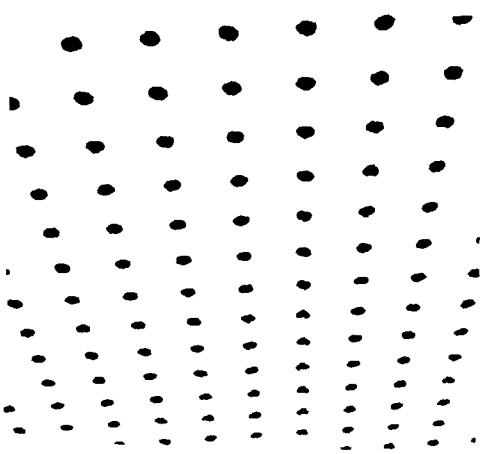
Figure 12. Muslims at a mosque.











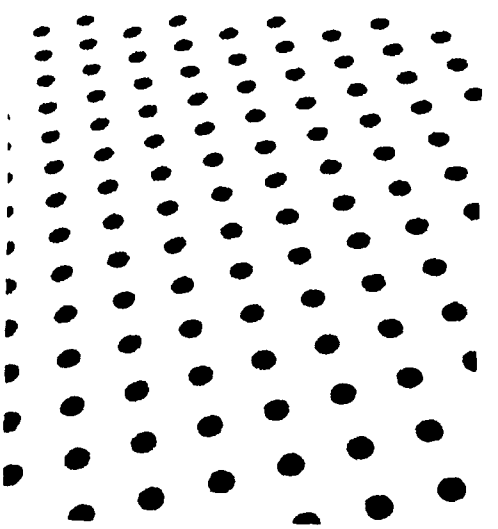
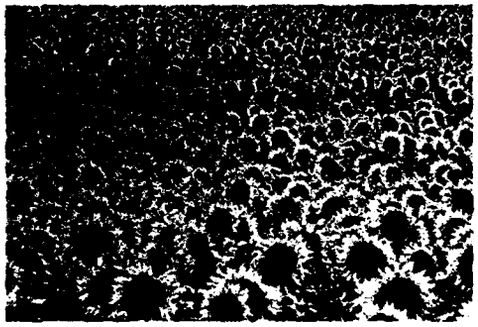
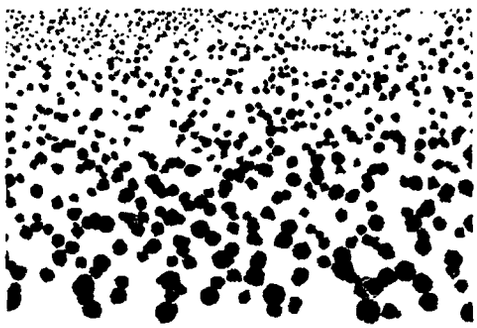


Figure 13. Fleecy clouds.

Figure 14. Audience at a 3D movie.





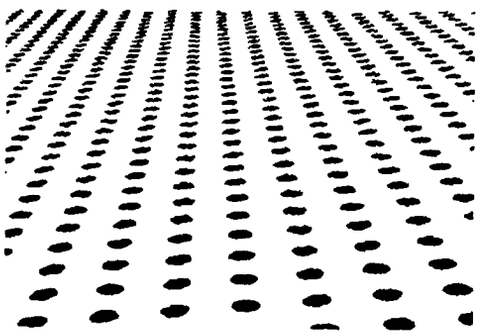
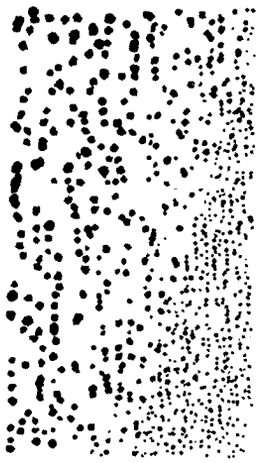


Figure 15. Sunflowers.



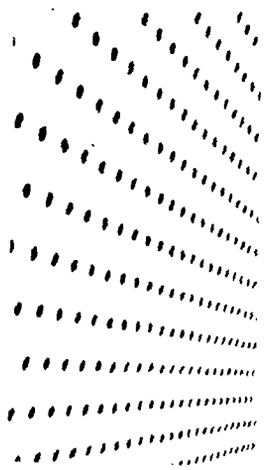
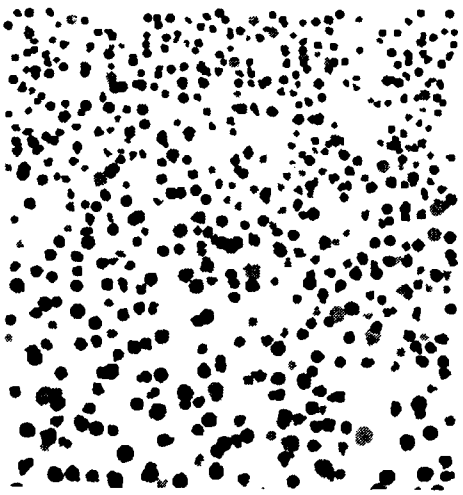


Figure 16. Tree trunk.



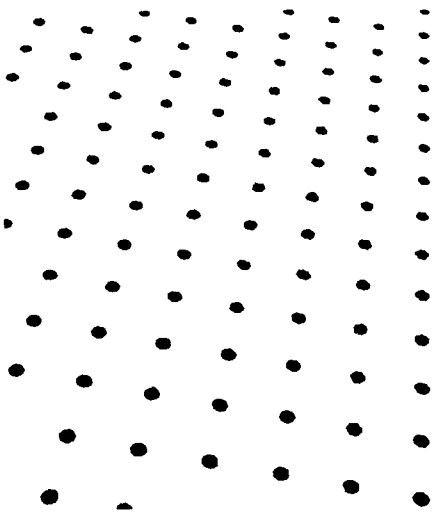
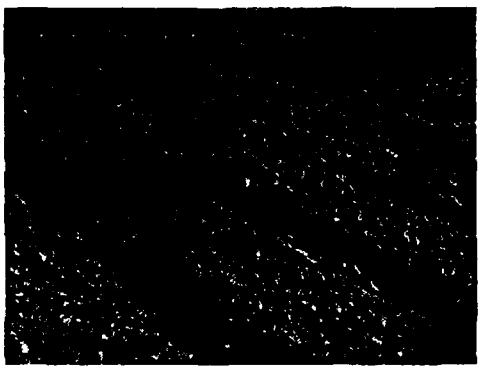
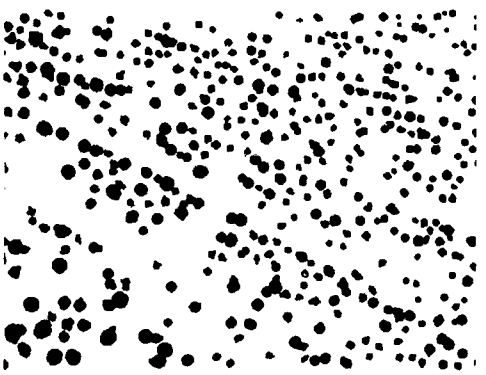


Figure 17. Bathers on the Ganges.





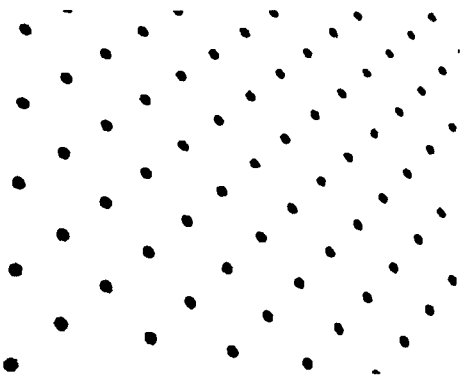
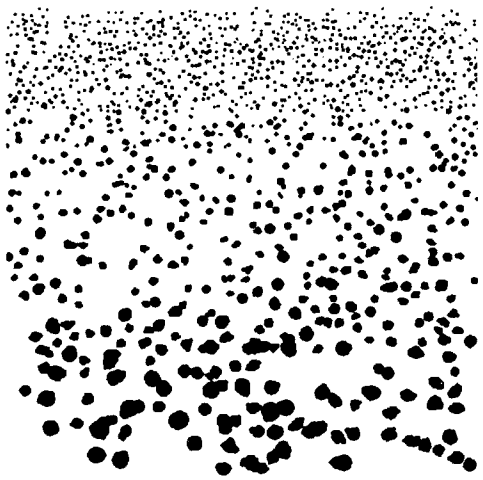


Figure 18. A plowed field.



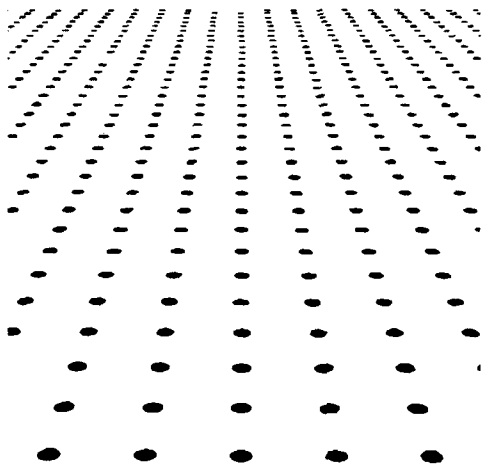
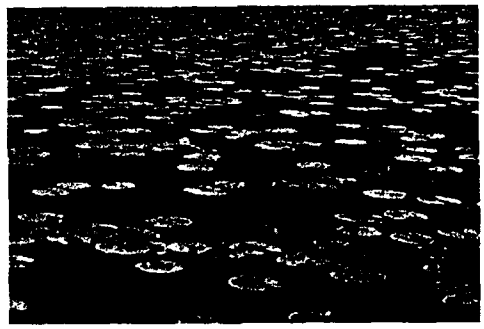
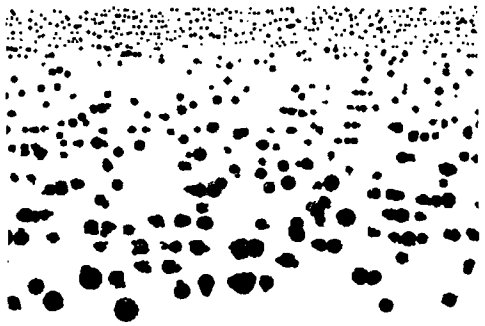


Figure 19. A field of flowers.





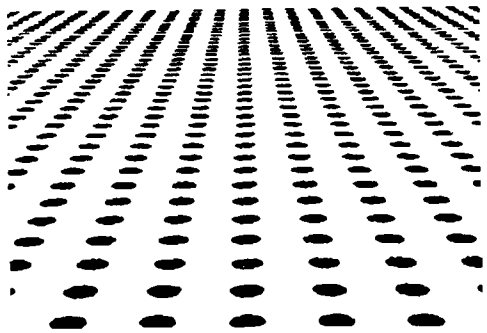
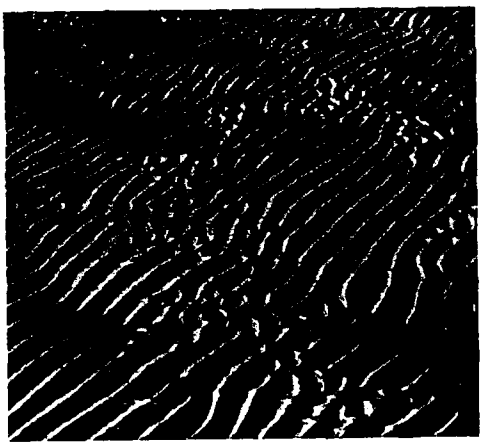


Figure 20. Water lillies.



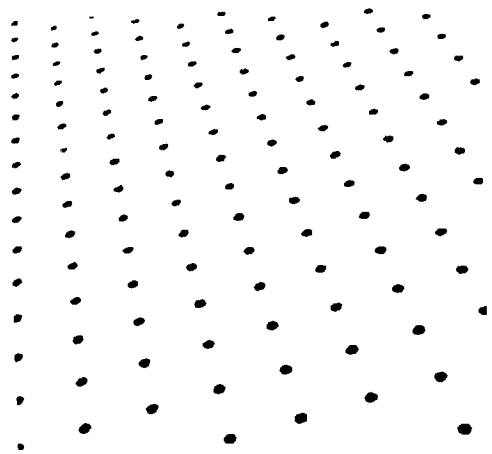
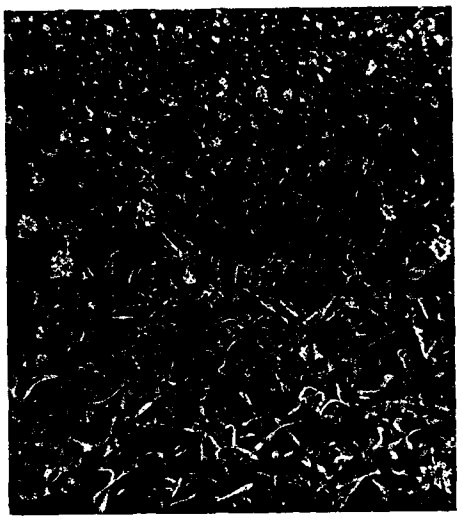
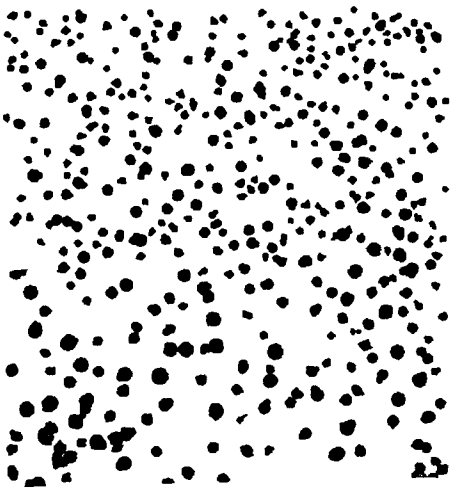


Figure 21. Ripple marks in a shallow sea.





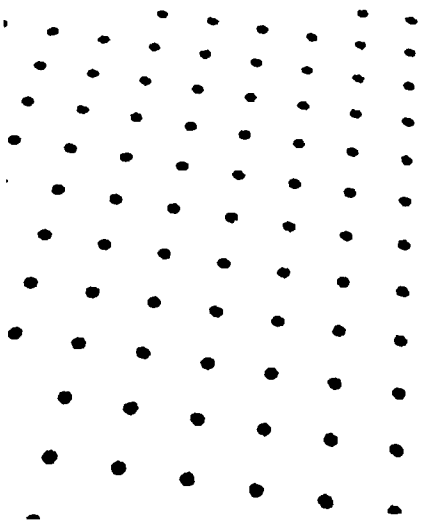
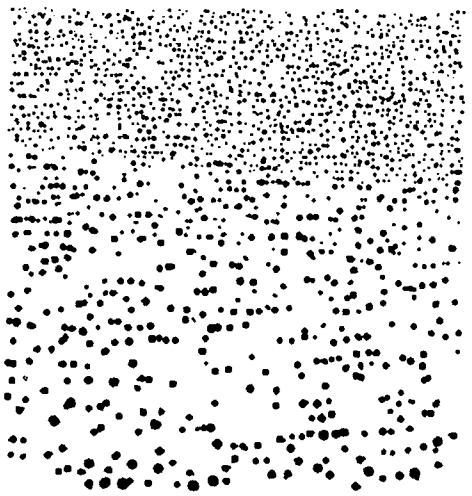


Figure 22. Water Hyacinths.



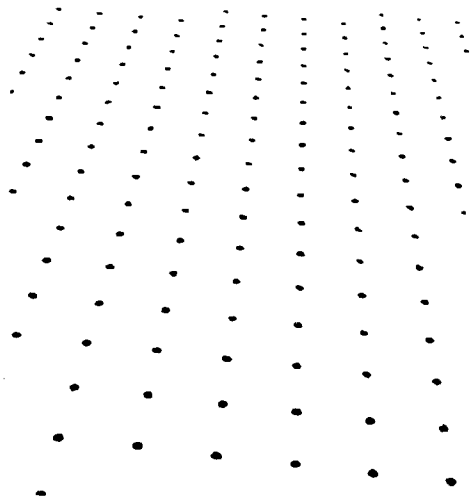
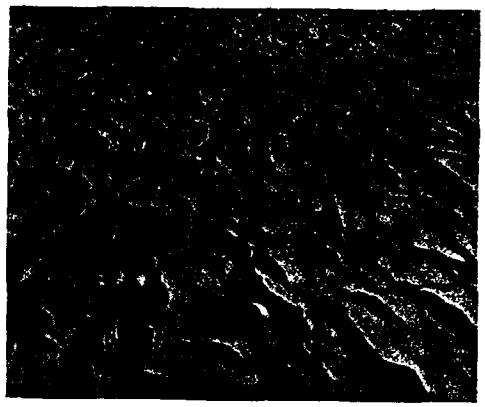
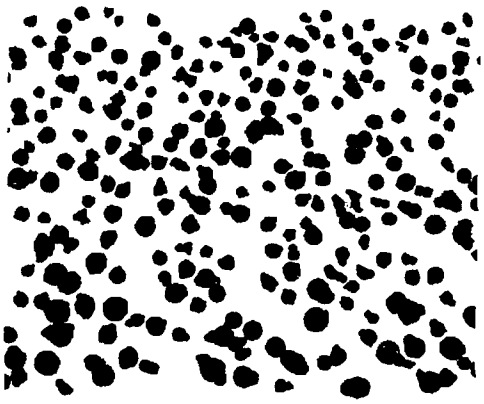


Figure 23. The Toulumne River.





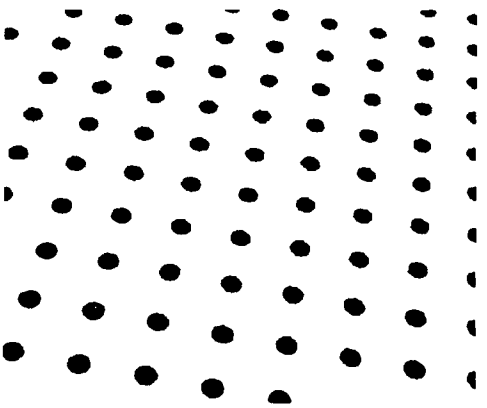


Figure 24. Sand by the Adriatic Sea.

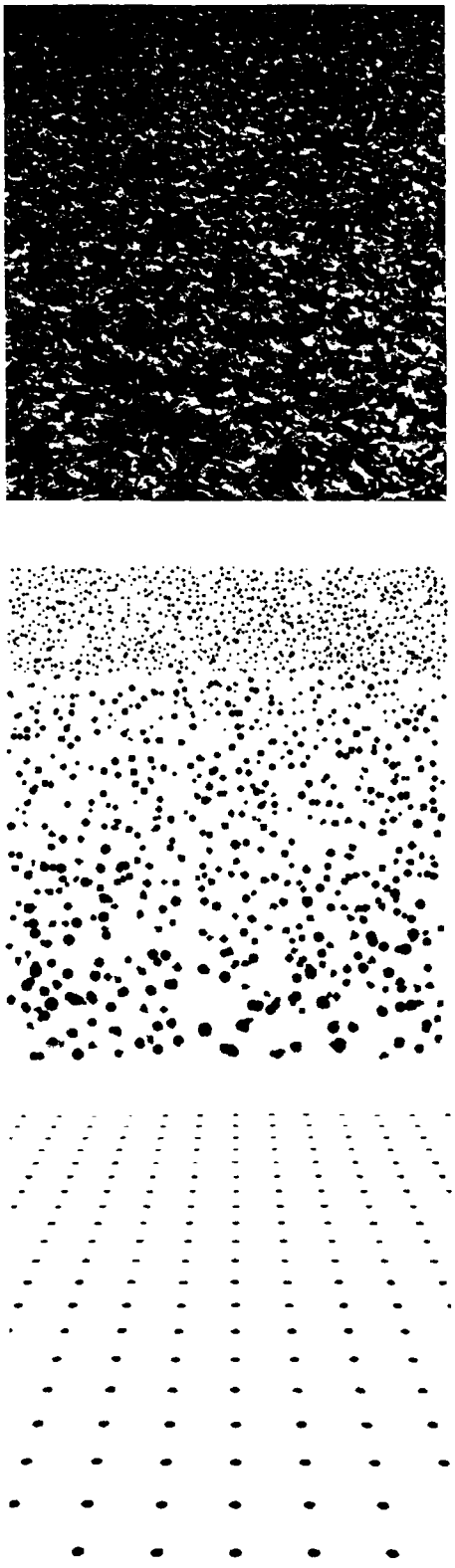


Figure 25. Fallen leaves.


## ACKNOWLEDGEMENTS

Parts of the research reported here were supported by grants from the National Scientific Foundation, Air Firce Office of Scientific Research, National Bureau of Standards, AT&T, Rockwell, Eastman Kodak, and Northrop. This support is gratefully acknowledged.

# Physically Based Modeling for Vision and Graphics

Andrew Witkin
Michael Kass
Demetri Terzopoulos
Kurt Fleischer

*Schlumberger Palo Alto Research*
3340 Hillview Avenue, Palo Alto, CA 94304

## I. Overview

A few years ago the Vision and Modeling group at SPAR began working on unified solutions to problems in computer vision and computer graphics, motivated by the observation that "vision is backwards graphics." The effort has proven fruitful, in some ways even more so than we anticipated. As it turns out, the benefits of combining vision and graphics are substantial and concrete: we've reached a point at which we are able to use the same approach, the same mathematics, in fact, the same *code* to solve a wide range of interesting problems in vision, graphics, animation, motion control and planning, and design.

Surprisingly, one reason for the progress we've made is that the dictum that "vision is backwards graphics" turns out to be *wrong*. In fact, graphics is backwards too. It is true that image synthesis runs the laws of optics forward to get images from models, while visual analysis in the "shape-from-x" school seeks to run the same laws backwards to get models from images. However, as people in the graphics community tired of seeing exquisitely ray-traced images of simple objects floating passively in space, the cutting edge in graphics research underwent a marked shift from image synthesis to *model* synthesis, focusing on the economical creation of realistic complex forms and natural coordinated motion.

The key problems in model synthesis are inverse problems in which a compact specification of the constraints that a shape or motion must fulfill is converted into an explicit description of the shape or motion itself. Viewed this way, model synthesis in graphics differs from visual shape and motion analysis primarily in the source of the constraints. In graphics, the constraints are supplied by a user, encoding desiderata, while in vision they encode evidence extracted from the image data.

Our approach to solving both sets of problems uses the machinery of physics. Our models are composed of simulated materials that move and deform in response to applied forces. Constraints, whether derived from images or specified by a human model-builder, are imposed by applying carefully designed constraint forces to the models. These forces serve to mold the model into a form that satisfies the constraints they represent, vanishing only when the constraints are met. For instance, a constraint that given points on two objects must coincide may be treated as a stiff spring that pulls them together. A constraint that a surface point must *project* to a particular image point is a spring pulling the surface model toward the *ray* corresponding to the image point. The manner in which a model responds to these and other applied forces is used to encode expected or desired properties of the object, such as a tendency toward smoothness. Additional forces, applied dynamically by the user, provide a remarkably effective mode of user interaction. Mathematically, this approach yields systems of ordinary or partial differential equations governing the evolution of the model through time.

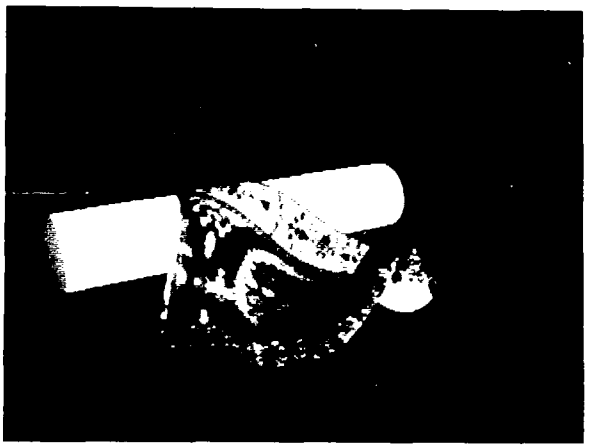


Figure 1: A single frame from an animation sequence of a rug draping over other objects

This section presents a brief overview, more in logical than chronological order, of the results that this approach has yielded. Each of the three following sections will present a selected topic in greater technical detail.

**Animating elastic objects.** Deformable models are free-form curves, surfaces, and solids imbued with internal energy functions that make them respond actively to applied forces. The most conceptually straightforward application of these models is to the direct simulation of non-rigid bodies such as paper, cloth, and rubber (figure 1) as described in [41]. We have used deformable models as a means of animating wind-blown flags, flying carpets, tearing cloth [14], and bouncing vegetables [13] among other things.

**Computer modeling clay.** The same machinery that produces animation of flags waving in the wind may be used as a powerful model-building medium, a kind of modeling clay in which free-form shapes are created by pushing and pulling on the deformable model interactively, nailing it down explicitly at important points, subjecting it to shape-controlling force fields, and so forth [40]. We are currently experimenting with this approach for real-time surface modeling using a stereo display and 3D pointing device.

**Snakes.** The idea of elastic media as model-building tools applies to visual analysis as well. A snake [24] is a simulated piece of stretchy, springy wire that responds in real–time to a variety of applied forces. We immerse snakes in force fields derived from images to create an easy and accurate means of interactively localizing edges and other line features. The image forces rapidly pull the snake towards the nearest feature of interest. If the snake gets caught in an undesired local energy minimum, the user can easily push it towards the desired minimum by exerting forces controlled by a mouse. The snake's internal strain energy provides a preference for smoothness, allowing it to complete contours over regions of low contrast. As a result, snakes can "see" subjective contours just as well as traditional lines and edges. Snakes also prove to be effective motion trackers—as the image changes, the energy wells move and the snake follows. A detailed description is presented in Section 2.

**Analysis of oriented patterns.** Elastic models have also proven useful for analyzing oriented or striated textures. In [23, 47], an energy function is used to align the rectangular grid of parameter lines on an elastic sheet to the locally measured orientation of an image. The deformed grid defines a coordinate system in which the oriented pattern is "straightened out," creating a more nearly stationary texture that is far simpler to analyze than the original. Figure 2 shows the straightening process applied to a wood texture.

**Scale space image matching.** We have applied elastic sheets to the problem of computing stereo and motion correspondence.[46] The images to be matched are mapped onto elastic sheets which are deformed by a force arising from a simple local correlation measure. To avoid undesired local energy minima, we use *scale-space continuation* in which the energy function is blurred, a large-scale minimum found, and the minimum tracked through continuous deblurring. The blurring continuum is created by simultaneously varying the elastic stiffness of the sheet, the spatial extent of the correlation measure and the blurring of the images.

**Symmetry-seeking models and 3D reconstruction.** In [42] we created symmetry-seeking computer modeling clay. The simulated material prefers axisymmetric shapes, but can be pulled away from symmetry by applied forces. We immerse this symmetry-seeking material in a force field that coerces it into a 3D shape whose 2D silhouette matches an edge in the image. The result is a full 3D reconstruction of the shape. When stereo images are available, we can simultaneously coerce the model to match silhouettes in both images. As with snakes, the symmetry-seeking model tracks motion by following the moving energy wells in an image sequence [43]. Figure 3 contrasts the full 3D description computed by by the symmetry-seeking model with the 2.5D description computed by the scale-space stereo technique. A detailed description of the symmetry-seeking model is provided in Section 3.

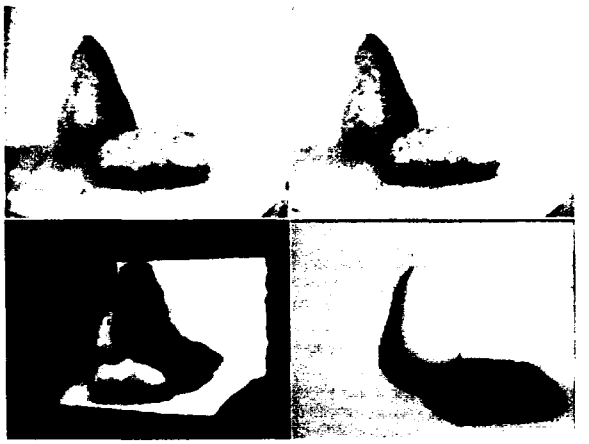


Figure 3: Top: Stereo images of a pear and potato. *Lower-left:* 2.5D depth map from scale-space stereo. The original image has been mapped onto the 2.5D surface and shown from a different viewpoint. Parts of the scene not visible from the original viewpoint are left black. Lower-right: 3D symmetric models. Unlike the 2.5D model, this model looks edible from behind.

**Self-assembling parameterized models.** The methods described so far employ free-form deformable models. A different but equally useful style of modeling creates complex shapes and structures by the composition of simple building blocks—basic geometric shapes such as cylinders and blocks, subjected to transformation operators such as translations, rotations, bends, twists, subtraction, etc. The methods described above apply equally well to this alternative modeling scheme. In [48] we apply energy methods to hierarchic parameterized models as an efficient means of constructing and animating models of complex objects. Figure 4 shows the parts of a model moving and deforming to satisfy attachment constraints.

**Visual analysis using parameterized models.** The same method can be applied to visual interpretation by deriving constraint forces from images. We are currently working on methods that reconstruct shapes and motion by attracting critical curves and points to rays derived from image features. Interestingly, the method is not restricted to the recovery of shape. We have used the method suc-

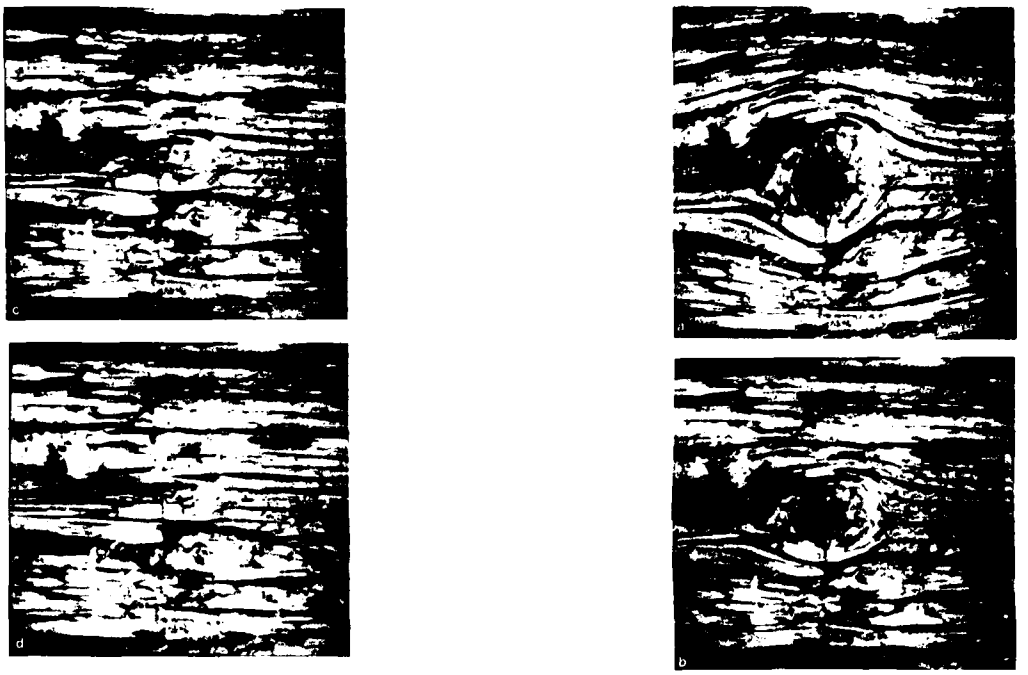


Figure 2: A natural wood texture (upper left) is transformed into a "straightened" pattern (lower right) in which the knot is eliminated and the curved grain lines become straight.

cessfully to match camera, lighting, and surface reflectance models, as well as shape, to real scenes.

**Cooking with Kurt.** "Cooking with Kurt" [13] is a computer video that intimately combines live and synthetic images, using nearly all of the techniques described above. The story centers on some vegetables sitting on a kitchen table that come to life and dance around. We began with live video including real vegetables. To animate them, it was necessary to obtain a sufficiently accurate model of the scene—the shapes and colors of the vegetables, as well as the geometry of the surfaces with which they interact, the viewing geometry, and the illumination—that we could undetectably excise the real objects from the scene, and matte in our synthetic animate copies. To do this, we used snakes to extract silhouettes of the vegetables, symmetry-seeking models to reconstruct their 3D shapes, and parameterized models to match their surface reflectance functions Parameterized models were also used to match the major surrounding surfaces, the camera parameters, and the illuminants. The resulting scene model allowed us not only to synthesize accurate images of our subjects, but have them cast correct shadows on the surrounding real objects. Once the substitution was made, we used elasticity and energy constraints to animate the vegetables, modeling them as elastic water-bags equipped with gyroscopes to maintain balance, and able to bounce on the floor as a means of locomotion.

**Spacetime constraints.**

Our most recent work [50] addresses the problem of planning and executing complex, physically accurate constrained motions. The basic idea of spacetime constraints is to represent the state of a physical system—the variables denoting positions and time-dependent forces—explicitly over the entire time interval of interest. We do this by replicating the instantaneous state of the system over a number of time steps, permitting time-derivatives—velocities and accelerations—to be computed as finite differences. We then define the motion to be performed by constraining objects' positions and such at various times, and providing optimization criteria, for instance involving mechanical efficiency or smoothness of motion, to specify how the motion should be performed within the hard constraints. We then impose *physics* as an additional constraint of the form $f = ma = 0$, and solve the resulting constrained optimization problem. The solution is a physically valid motion that optimally satisfies the constraints .

Spacetime constraints provide an enormously powerful mechanism for synthesizing realistic, complex, coordinated motions just by stating in bare essentials what the motion is supposed to accomplish. We believe that the spacetime method has great potential for visual motion interpretation as well and are currently working to tap this potential. We hope, among other things, to use the method to recover not only 3D geometry but also *physics*, using the motion observed in the image to solve for force and mass. The method is described in detail in Section 4.

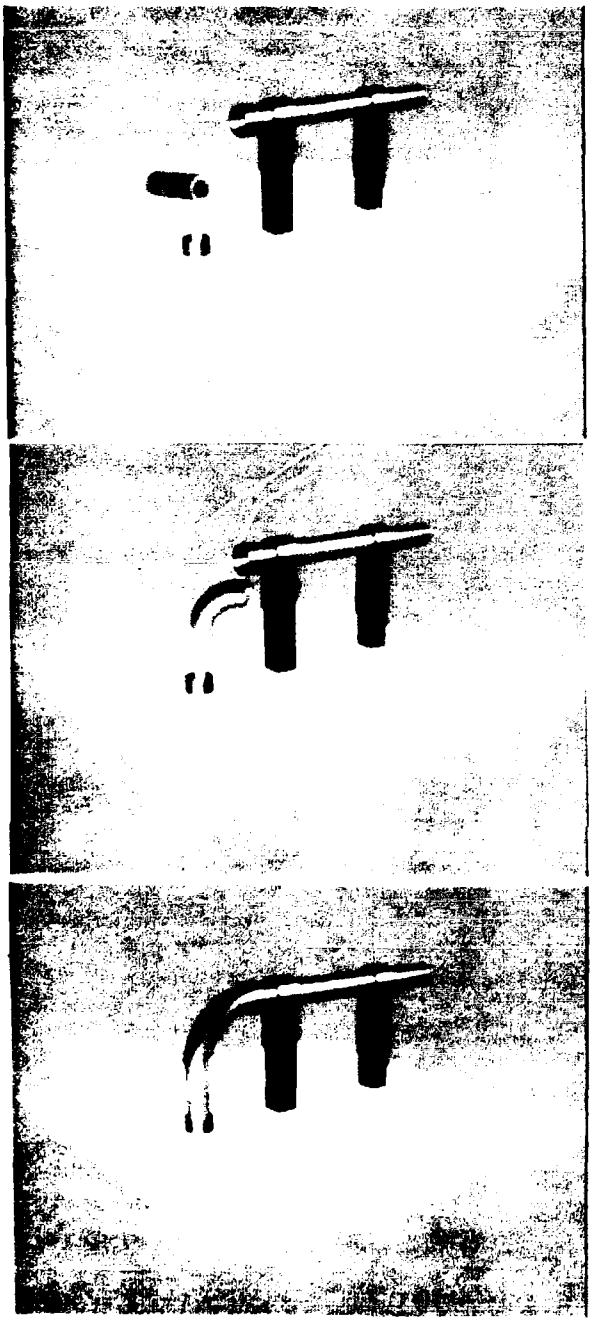


Figure 4: A flexible pipe attaching itself to neighboring pipes under the influence of constraint forces. The initial configuration is shown at the top, the final assembled configuration at the bottom, and an intermediate one in the middle.

## II. Snakes: Active contour models[1]

In recent computational vision research, low-level tasks such as edge or line detection, stereo matching, and motion tracking have been widely regarded as autonomous bottom-up processes. Marr and Nishihara[28], in a strong statement of this view, say that up to the 2.5D sketch, "no 'higher-level' information is yet brought to bear: the computations proceed by utilizing only what is available in the image itself." This rigidly sequential approach propagates mistakes made at a low level without opportunity for correction. It therefore imposes stringent demands on the reliability of low-level mechanisms. As a weaker but more attainable goal for low-level processing, we argue that it ought to provide sets of alternative organizations among which higher level processes may choose, rather than shackling them prematurely with a unique answer.

In this paper we investigate the use of energy minimization as a framework within which to realize this goal. We seek to design energy functions whose local minima comprise the set of alternative solutions available to higher level processes. The choice among these alternatives could require some type of search or high-level reasoning. In the absence of a well developed high-level mechanism, however, we use an interactive approach to explore the alternative organizations. By adding suitable energy terms to the minimization, it is possible for a user to push the model out of a local minimum towards the desired solution. The result is an active model that falls into the desired solution when placed near it.

Energy minimizing models have a rich history in vision going back at least to Sperling's stereo model [38]. Such models have typically been regarded as autonomous, but we have developed interactive techniques for guiding them. Interacting with such models allows us to explore the energy landscape very easily and develop effective energy functions which have few local minima and little dependence on starting points. We hope thereby to make the job of high-level interpretation manageable yet not constrained unnecessarily by irreversible low-level decisions.

The problem domain which we address is that of finding salient image contours - edges, lines, and subjective contours - as well as tracking those contours during motion and matching them in stereopsis. Our variational approach to finding image contours differs from the traditional approach of detecting edges and then linking them. In our model, issues such as the connectivity of the contours and the presence of corners affect the energy functional and hence the detailed structure of the locally optimal contour. These issues can, in principle, be resolved by very high-level computations. Perhaps more importantly, high-level mechanisms can interact with the contour model by pushing it towards an appropriate local minimum. Optimization and relaxation have been used previously in edge

[1] The material in this section is adapted from the paper "Snakes: Active Contour Models," by Michael Kass, Andrew Witkin, and Demetri Terzopoulos, which appears in *International Journal of Computer Vision*, 1 (4), 1987.

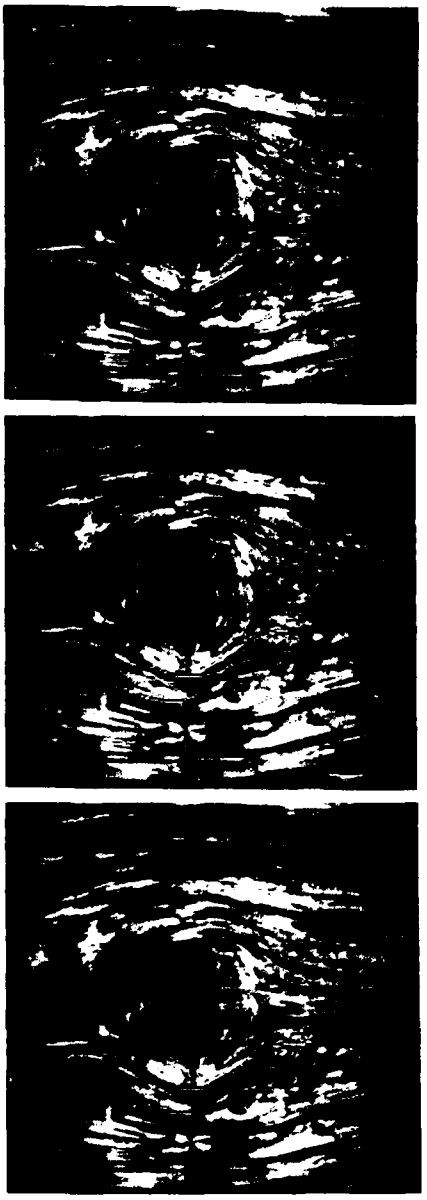


Figure 5: Top: Original wood photograph from Brodatz. Others: Two different local minima for the active contour model.

and line detection [9, 11, 32, 54, 55], but without the interactive guiding used here.

In many image interpretation tasks, the correct interpretation of low-level events can require high-level knowledge. Consider, for example, the three perceptual organizations of two dark lines in figure 5. The three different organizations correspond to three different local minima in our line-contour model. It is important to notice that the shapes of the lines are materially different in the three examples, not just because of a different linking of line segments. The segments themselves are changed by the perceptual organization.

Without detailed knowledge about the object in view, it is difficult to justify a choice among the three interpretations. Knowing that wood is a layered structure, or perhaps inferring its layered structure from elsewhere in the picture could help to rule out interpretation (b). Beyond that, the "correct" interpretation could be very task dependent. In many domains, such as analyzing seismic data, the choice of interpretation can depend on expert knowledge. Different seismic interpreters can derive significantly different perceptual organizations from the same seismic sections depending on their knowledge and training. Because a single "correct" interpretation cannot always be defined, we suggest low-level mechanisms which seek appropriate local minima instead of searching for global minima.

Unlike most other techniques for finding salient contours, our model is *active*. It is always minimizing its energy functional and therefore exhibits dynamic behavior. Because of the way the contours slither while minimizing their energy, we call them snakes. Changes in high-level interpretation can exert forces on a snake as it continues its minimization. Even in the absence of such forces, snakes exhibit hysteresis when exposed to moving stimuli.

Snakes do not try to solve the entire problem of finding salient image contours. They rely on other mechanisms to place them somewhere near the desired contour. However, even in cases where no satisfactory automatic starting mechanism exists, snakes can still be used for semi-automatic image interpretation. If an expert user pushes a snake close to an intended contour, its energy minimization will carry it the rest of the way. The minimization provides a "power assist" for a person pointing to a contour feature.

Snakes are an example of a more general technique of matching a deformable model to an image by means of energy minimization. In spirit and motivation, this idea shares much with the rubber templates of Widrow[52]. From any starting point, the snake deforms itself into conformity with the nearest salient contour. We have applied the same basic techniques to the problem of 3D object reconstruction from silhouettes by using energy minimizing surfaces with preferred symmetries [42]. We expect this general approach will find a wide range of applicability in vision.

## A. Basic Snake Behavior

Our basic snake model is a *controlled continuity*[39] spline under the influence of image forces and external constraint forces. The internal spline forces serve to impose a piecewise smoothness constraint. The image forces push the snake towards salient image features like lines, edges, and subjective contours. The external constraint forces are responsible for putting the snake near the desired local minimum. These forces can, for example, come from a user interface, automatic attentional mechanisms or high level interpretations.

Representing the position of a snake parametrically by $\mathbf{v}(s) = (x(s), y(s))$, we can write its energy functional as

$$\begin{aligned} E^*_{snake} &= \int_0^1 E_{snake}(\mathbf{v}(s))ds \\ &= \int_0^1 E_{int}(\mathbf{v}(s)) + E_{image}(\mathbf{v}(s)) \\ &\quad + E_{con}(\mathbf{v}(s))ds \end{aligned} \tag{1}$$

where $E_{int}$ represent the internal energy of the spline due to bending, $E_{image}$ gives rise to the image forces and $E_{con}$ gives rise to the external constraint forces. In this section, we develop $E_{int}$ and give examples of $E_{con}$ for interactive interpretation. $E_{image}$ is developed in section II-C.

## B. Internal Energy

The internal spline energy can be written

$$E_{int} = (\alpha(s)|\mathbf{v}_s(s)|^2 + \beta(s)|\mathbf{v}_{ss}(s)|^2)/2 \tag{2}$$

The spline energy is composed of a first order term controlled by $\alpha(s)$ and a second order term controlled by $\beta(s)$. The first-order term makes the snake act like a membrane and the second-order term makes it act like a thin plate. Adjusting the weights $\alpha(s)$ and $\beta(s)$ controls the relative importance of the membrane and thin plate terms. Setting $\beta(s)$ to zero at a point allows the snake to become second-order discontinuous and develop a corner. The controlled continuity spline is a generalization of a Tikhonov stabilizer [44] and can formally be regarded as *regularizing* [35, 36] the problem.

Details of our minimization procedure are given in [24]. The procedure is an $O(n)$ iterative technique using sparse matrix methods. Each iteration effectively takes implicit Euler steps with respect to the internal energy and explicit Euler steps with respect to the image and external constraint energy. The numeric considerations are relatively important. In a fully explicit Euler method, it takes $O(n^2)$ iterations each of $O(n)$ time for an impulse to travel down the length of a snake. The resulting snakes are flaccid. In order to erect more rigid snakes, it is vital to use a more stable method that can accommodate the large internal forces. Our semi-implicit method allows forces to travel the entire length of a snake in a single $O(n)$ iteration.

**Snake Pit.** In order to experiment with different energy functions for low-level visual tasks, we have developed a user-interface for snakes on a Symbolics Lisp Machine. The interface allows a user to select starting points and exert forces on snakes interactively as they minimize their energy. In addition to its value as a research tool, the user-interface has proven very useful for semi-automatic image interpretation. In order to specify a particular image feature, the user has only to push a snake near the feature. Once close enough, the energy minimization will pull the snake in the rest of the way. Accurate tracking of contour features can be specified in this way with little more effort than pointing. The snake energy minimization provides a "power assist" for image interpretation.

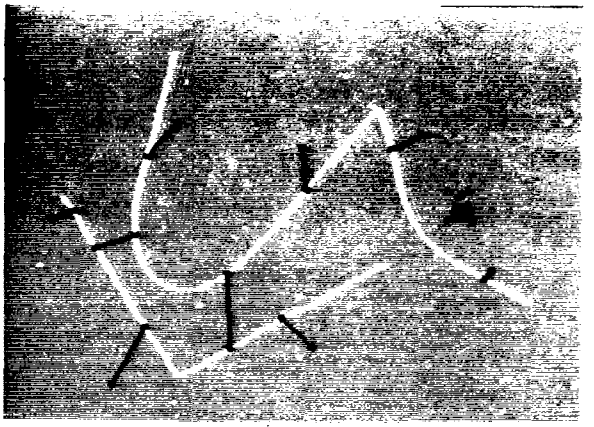


Figure 6: The Snake Pit user-interface. Snakes are shown in black, springs and the volcano are in white.

Our interface allows the user to connect a spring to any point on a snake. The other end of the spring can be anchored at a fixed position, connected to another point on a snake, or dragged around using the mouse. Creating a spring between $\mathbf{x}_1$ and $\mathbf{x}_2$ simply adds $-k(\mathbf{x}_1 - \mathbf{x}_2)^2$ to the external constraint energy $E_{con}$.

In addition to springs, the user interface provides a $1/r^2$ repulsion force controllable by the mouse. The $1/r$ energy functional is clipped near $r = 0$ to prevent numerical instability, so the resulting potential is depicted by a volcano icon. The volcano is very useful for pushing a snake out of one local minimum and into another.

Figure 6 shows the snake-pit interface being used. The two dark lines are different snakes which the user has connected with two springs shown in white. The other springs attach points on the snakes to fixed positions on the screen. In the upper right, the volcano can be seen bending a nearby snake. Each of the snakes has a sharp corner which has been specified by the user.

## C. Image Forces

In order to make snakes useful for early vision we need energy functionals which attract them to salient features in images. In this section, we present three different energy functionals which attract a snake to lines, edges, and terminations. The total image energy can be expressed as a weighted combination of the three energy functionals

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term}. \tag{3}$$

By adjusting the weights, a wide range of snake behavior can be created.

**Line Functional.** The simplest useful image functional is the image intensity itself. If we set

$$E_{line} = I(x, y) \tag{4}$$

then depending on the sign of $w_{line}$, the snake will be attracted either to light lines or dark lines. Subject to its other constraints, the snake will try to align itself with the lightest or darkest nearby contour. This energy functional was used with the snakes shown in figure 5. By pushing with the volcano, a user can rapidly move a snake from one of these positions to another. The coarse control necessary to do so suggests that symbolic attentional mechanisms might be able to guide a snake effectively.

**Edge Functional.** Finding edges in an image can also be done with a very simple energy functional. If we set $E_{edge} = -|\nabla I(x,y)|^2$, then the snake is attracted to contours with large image gradients. An example of the use of this functional is shown in figure 7. In the upper left, a user has placed two snakes on the edges of the pear and potato. He has then pulled part of the snake off the pear with a spring. The remaining pictures show what happens when he lets go. The snake snaps back rapidly to the boundary of the pear.

**Scale Space.** In figure 7, the snake was attracted to the pear boundary from a fairly large distance away because of the spline energy term. This type of convergence is rather common for snakes. If part of a snake finds a low-energy image feature, the spline term will pull neighboring parts of the snake towards a possible continuation of the feature. This effectively places a large energy well around a good local minimum. A similar effect can be achieved by spatially smoothing the edge- or line- energy functional. One can allow the snake to come to equilibrium on a very blurry energy functional and then slowly reduce the blurring. The result is minimization by scale-continuation[45, 46].

In order to show the relationship of scale-space continuation to the Marr-Hildreth theory of edge-detection[30], we have experimented with a slightly different edge functional. The edge energy functional is

$$E_{line} = -(G_\sigma * \nabla^2 I)^2 \tag{5}$$

where $G_\sigma$ is a Gaussian of standard deviation $\sigma$. Minima of this functional lie on zero-crossings of $G_\sigma * \nabla^2 I$ which define edges in the Marr-Hildreth theory. Adding this energy term to a snake means that the snake is attracted to zero-crossings, but still constrained by its own smoothness. Figure 8 shows scale-space continuation applied to this energy functional. The upper left shows the snake in equilibrium at a very coarse scale. Since the edge energy function is very blurred, the snake does a poor job of localizing the edge, but is attracted to this local minimum from very far away. Slowly reducing the blurring leads the snake to the position shown in the upper right and finally to the position shown in the lower left. For reference, the zero-crossings of $G_\sigma * \nabla^2 I$ corresponding to the energy function of the snake in the lower left are shown superimposed on the same snake in the lower right. Note that the snake jumps from one piece of a zero-crossing contour to another. At this scale, the shapes of the zero-crossings are dominated by the small-scale texture rather than the region boundary, but the snake nevertheless is able to use the zero-crossings for localization because of its smoothness constraint.

## D. Termination Functional

In order to find terminations of line segments and corners, we use the curvature of level lines in a slightly smoothed image. Let $C(x,y) = G_\sigma(x,y) * I(x,y)$ be a slightly smoothed version of the image. Let $\theta = \tan^{-1}(C_y/C_x)$ be the gradient angle and let $\mathbf{n} = (\cos\theta, \sin\theta)$ and $\mathbf{n}_\perp = (-\sin\theta, \cos\theta)$ be unit vectors along and perpendicular to the gradient direction. Then the curvature of the level contours in $C(x,y)$ can be written

$$E_{term} = \frac{\partial\theta}{\partial\mathbf{n}_\perp} \tag{6}$$

$$= \frac{\partial^2 C/\partial\mathbf{n}_\perp^2}{\partial C/\partial\mathbf{n}} \tag{7}$$

$$= \frac{C_{yy}C_x^2 - 2C_{xy}C_xC_y + C_{xx}C_y^2}{(C_x^2 + C_y^2)^{3/2}} \tag{8}$$

By combining $E_{edge}$ and $E_{term}$, we can create a snake which is attracted to edges or terminations. Figure 9 shows an example of such a snake exposed to a standard subjective contour illusion[22]. The shape of the snake contour between the edges and lines in the illusion is entirely determined by the spline smoothness term. The variational problem solved by the snake is very closely related to a variational formulation proposed by Brady et al. [7] for the interpolation of subjective contours. Ullman's [51] proposal of interpolating using piecewise circular arcs would probably also produce a very similar interpolation. An appealing aspect of the snake model is that the same snake that finds subjective contours can very effectively find more traditional edges in natural imagery. It may, moreover, provide some insight into why the ability to see subjective contours is important.

## E. Motion

Once a snake finds a salient visual feature, it "locks on." If the feature then begins to move slowly, the snake will simply track the same local minimum. Movement which is too rapid can cause a snake to flip into a different local minimum, but for ordinary speeds and video-rate sampling, snakes do a good job of tracking motion. Figure 10 shows eight selected frames out of a two second video sequence. Edge-attracted snakes were initialized by hand on the speaker's lips in the first frame. After that, the snakes tracked the lip movements automatically.

The motion tracking was done in this case without any inter-frame constraints. Introducing such constraints will doubtless make the tracking more robust. A simple way to do so is to give the snake mass. Then the snake will predict its next position based on its previous velocity.

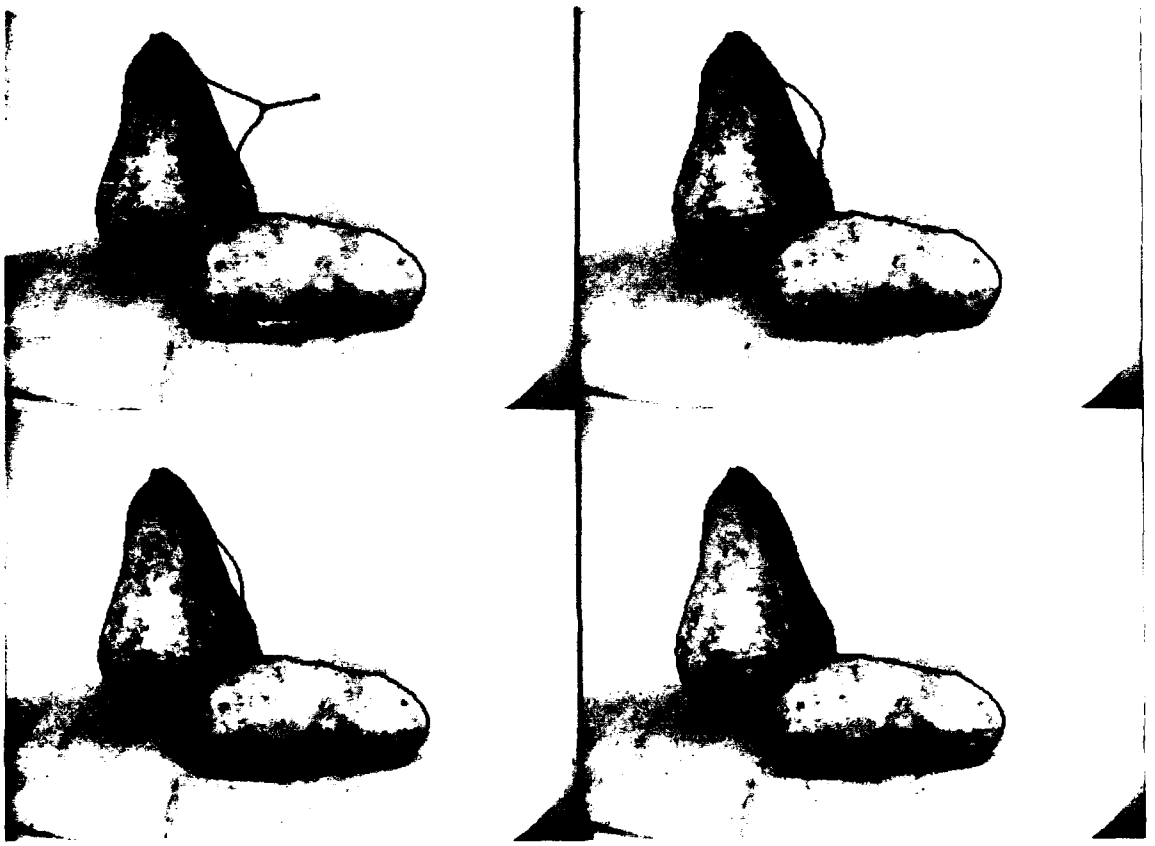


Figure 7: Two edge snakes on a pear and potato. Upper-left: The user has pulled one of the snakes away from the edge of the pear. Others: After the user lets go, the snake snaps back to the edge of the pear

## F. Conclusion

Snakes have proven useful for interactive specification of image contours. We have begun to use them as a basis for interactively matching 3D models to images. As we develop better energy functionals the "power assist" of snakes becomes increasingly effective. Scale-space continuation can greatly enlarge the capture region around features of interest.

The snake model provides a unified treatment to a collection of visual problems which in the past have been treated differently. Edges, lines, and subjective contours can all be found by essentially the same mechanisms. Tracking these features through motion and matching them in stereo is easily handled in the same framework.

Snakes, perhaps, embody Marr's notion of "least commitment" [31] more than his bottom-up 2.5D sketch. The snake provides a number of widely separated local minima to further levels of processing. Instead of committing irrevocably to a single interpretation, snakes can change their interpretation based on additional evidence from higher levels of processing. They can, for example, adjust monocular edge-finding based on binocular matches.

We believe that the ability to have all levels of visual processing influence the lowest-level visual interpretations will turn out to be very important. Local energy minimizing systems like snakes offer an attractive method for doing this. The energy minimization leaves a much simpler problem for higher level processing.

# III. Recovering 3D Shape, Depth, and Nonrigid Motion from profiles[2]

[2]The material in this section is adapted from the paper "Energy Constraints on Deformable Models: Recovering Shape and Nonrigid

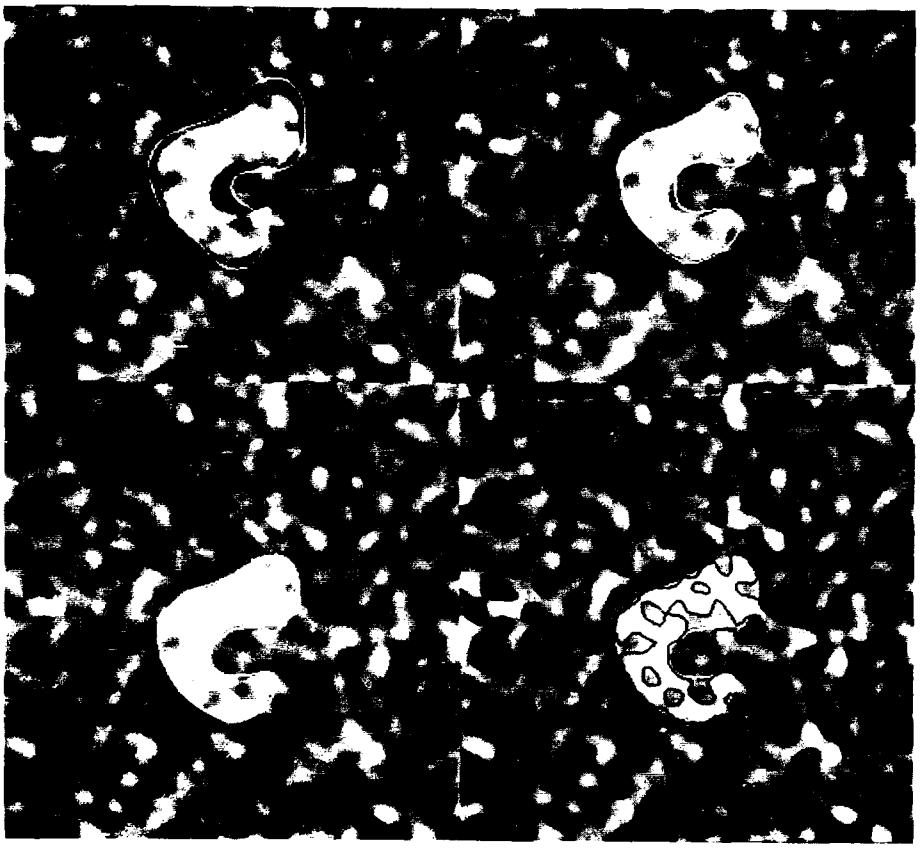


Figure 8: Upper-left: Edge snake in equilibrium at coarse scale. Upper-right: Snake in equilibrium at intermediate scale. Lower-left: Final snake equilibrium after scale-space continuation. Lower-right: Zero-crossings overlayed on final snake position.

The profile of an object, also known as its occluding contour, refers to the curve which outlines the image region covered by the optical projection of the object. The human visual system has the remarkable ability to infer the 3D shapes of objects from their 2D profiles in images. To emphasize this, David Marr was fond of showing Picasso's "Rites of Spring," which consists entirely of silhouettes (see figure 11 We present physically-based models with accompanying computational techniques capable of reconstructing, directly from the silhouettes, a 3D rendition of Picasso's "Rites" shown below.

Our approach, which is applicable to natural images, simplifies the integration of image information acquired from different viewpoints and/or different instants in time. Our models can infer the 3D shapes of objects in proper depth from their profiles in stereoscopic views. Moreover, they can capture the motion of objects moving nonrigidly in space by tracking the evolution of profiles in dynamic image sequences. Recovering shape, depth, and nonrigid motion of free-form, flexible objects from natural images—a task which the human visual system accomplishes routinely—has traditionally been considered a very difficult open problem in machine vision. Our algorithms can perform this task directly from image intensity data without requiring intermediate optic-flow fields or 2.5D surface representations.

To reconstruct models directly from natural images that possibly involve significant occlusions, we must exploit several powerful constraints in unison, some deriving from the sensory information content of images, others reflecting background knowledge about image formation. We take a physically-based modeling approach, in which objects are represented as elastically deformable bodies subject to continuum mechanical laws, while constraints are modeled as forces applied to the deformable bodies. Physically-based models integrate constraints in a natural way—by summing together the associated forces. The net force field propels the models through potentially complicated motions such that they satisfy the available constraints through time. We distinguish two types of forces: *Intrinsic forces* encode constraints internal to our deformable models. *Extrinsic forces* couple the models to the image

Motion" by Demetri Terzopoulos, Andrew Witkin, and Michael Kass, which first appeared in *Proc. AAAI-87*, Seattle. An extended version is in press in *Artificial Intelligence*.

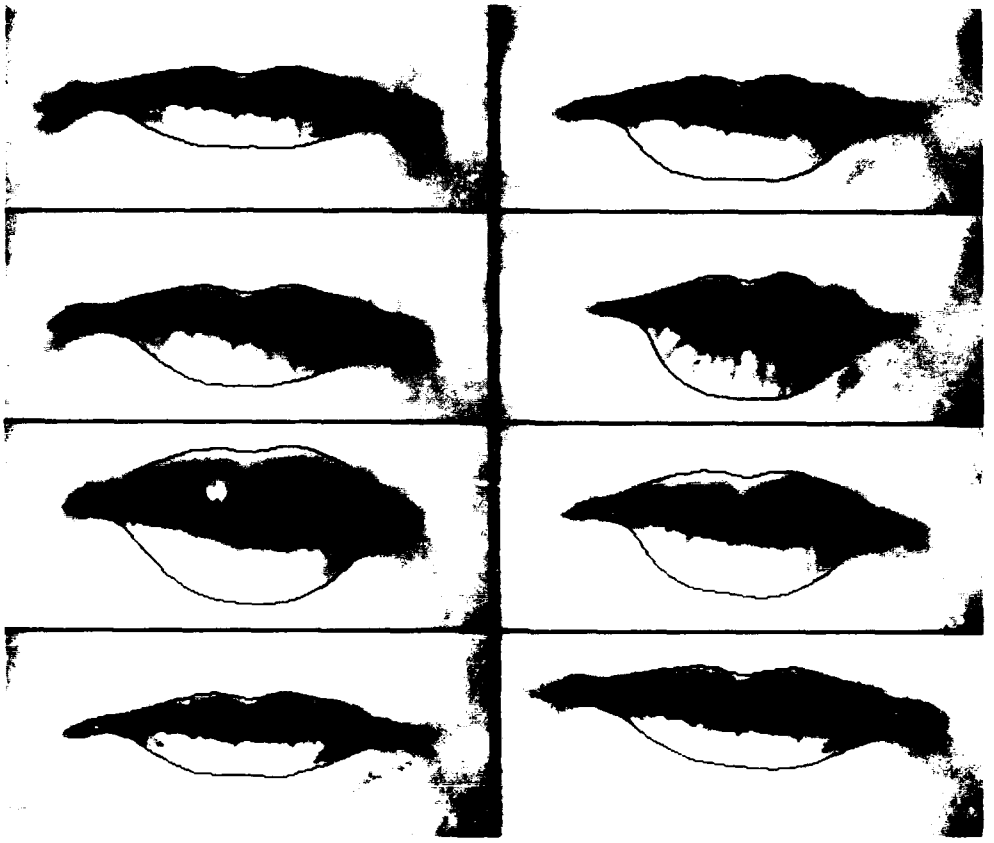


Figure 10: Selected frames from a 2 second video sequence showing snakes used for motion tracking. After being initialized to the speaker's lips in the first frame, the snakes automatically track the lip movements with high accuracy.

data and provide an avenue for user interaction.

The intrinsic constraints reflect generically valid assumptions about natural objects. Our deformable models apply a basic constraint—surface coherence—which can accurately quantify the free-form shapes and motions of natural objects. The constraint is inherent in the elastic forces prescribed by the physics of deformable continua. These forces elicit piecewise continuous deformations.

A second generic constraint built into our models is symmetric regularity, an attribute of many natural and synthetic objects. Rather than imposing strict symmetries through explicit parameterization, we design more liberal *symmetry-seeking* intrinsic forces. These forces constrain the deformations of the model in order to give it a preference for certain symmetries. Representing symmetry as constrained deformation rather than through geometric parameterization frees our model from the shackles of particular parametric shape families such as, say, the quadrics—spheres, cylinders, ellipsoids, etc.

Our work to date considers the reconstruction of the 3D shape and nonrigid motion of objects possessing approximate axial symmetry. Our model is close in spirit to the generalized cylinder representation first recommended in 1971 by Binford as a convenient description of 3D surfaces for the purposes of vision [5]. It captures axisymmetry much like the generalized cylinder; however, we take seriously the fact that many objects of interest are only approximately symmetric. Unlike generalized cylinders, our model can accommodate deviations from exact symmetry by deforming. Only as the intrinsic forces are strengthened does the symmetry-seeking model tend to impose exact symmetry. As the intrinsic forces are weakened, however, the model will be able to faithfully represent increasingly asymmetric shapes, although axisymmetric shapes have greater stability and hence are preferred.

The extrinsic constraints reflect, in part, observations about the environment that can be extracted from sensory data. Although, in principle, we can exploit within symmetry-seeking models a variety of image based cues, including shading and texture, for the time being we make exclusive use of information about profiles. The models are embedded in a force field which encodes the profile information. The ambient forces mold the deformable models to make their 3D shapes consistent with the observed 2D profiles of objects or subparts.

Another possible source of extrinsic constraints is a human operator. We augment the ambient force field with forces controlled by computer pointing devices, thereby

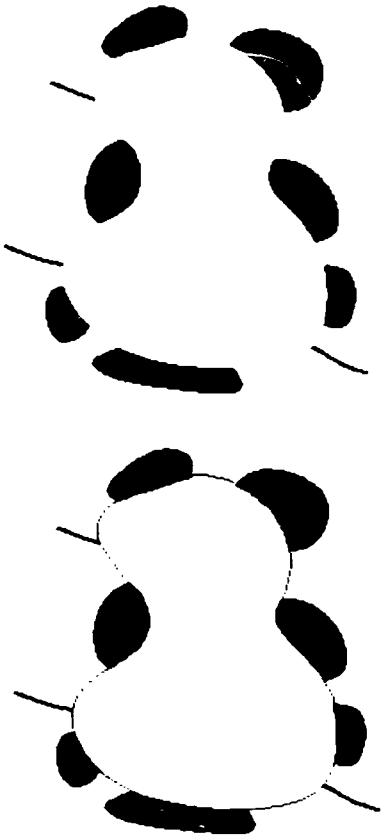


Figure 9: Top: Standard subjective contour illusion. Bottom: Edge/Termination snake in equilibrium on the subjective contour

Figure 11: 'Rites of Spring' by Pablo Picasso (top). "We immediately interpret such silhouettes in terms of particular three-dimensional surfaces—this despite the paucity of information in the image itself. In order to do this, we plainly must invoke certain *a priori* assumptions and constraints about the nature of the shapes." –D. Marr [27]. Silhouette shapes in confluence with *a priori* constraints intrinsic to the *symmetry-seeking models* described in this article can suffice to recover 3D shapes from a monocular image. (bottom) 3D reconstruction of Picasso's "Rites" using 43 symmetry-seeking models (see text). Note that the reconstructed shapes are only approximately axisymmetric, as dictated by the shapes of the silhouettes.

providing opportunity for a user to guide the reconstruction process. We can create symmetry-seeking models and pull or push them through space using a 3D interactive pointer (flying mouse, a.k.a. bat). We bring models near imaged objects of interest and we monitor them as wireframes projected into the image plane(s) while they capture the shapes of these objects. On a sufficiently powerful computer, deformable models offer a fully interactive modeling medium of practical interest in its own right. Nonetheless, our long-range goal in the context of machine vision is to replace the user input with fully automatic top-down control processes.

Figure 12 illustrates the reconstruction of a crooknecked squash from its monocular image using a symmetry-seeking model. The user initializes the model by specifying the projection of the spine in the image plane near the medial axis of the object. In the monocular case the model is subject to extrinsic forces expressed as the gradient of a potential function which measures the local contrast in the Gaussian-smoothed image. Hence, the high contrast contour in the image (by assumption, the profile of the object) attracts the occluding boundary of the model (the locus of points along which lines of sight graze the surface of the model). The model comes to equilibrium in the ambient force field such that its occluding boundary, relative to the viewpoint associated with the image, is consistent with the shape of the object profile in the image. The model's intrinsic continuity and symmetry-seeking forces specify 3D shape over the remainder of its surface, including hidden portions. Figure 13 sketches the

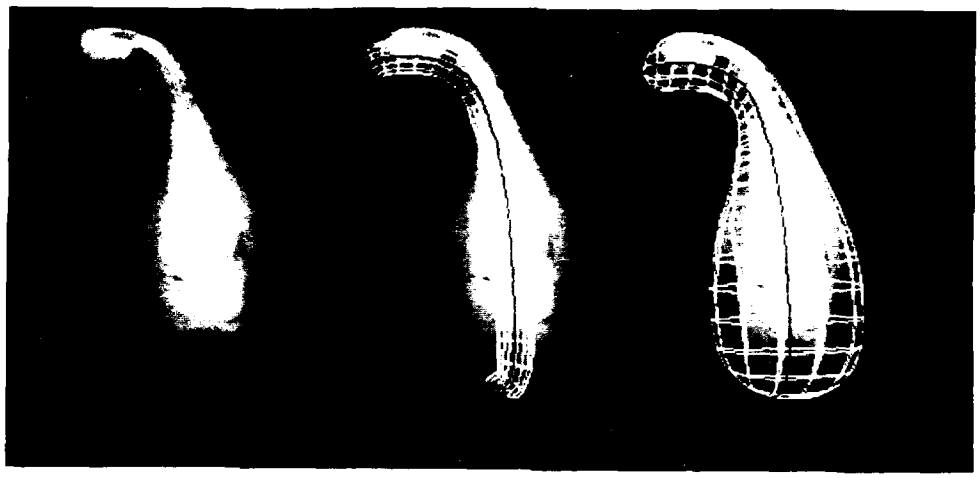


Figure 12: Reconstruction of a 3D symmetry-seeking model. (left) Squash image. (center) User initialized spine shown in (black curve) and initial tube (white wireframe). (right) Reconstructed model displayed as a wire-frames projected into the image.

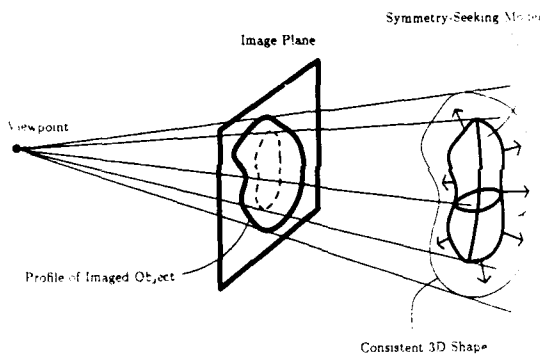



Figure 13: Monocular reconstruction scenario. The arrows depict extrinsic forces in space which act on the symmetry-seeking model's occluding boundary as seen from the viewpoint. The forces deform the 3D model so as to make its image plane projection (dotted curve) more consistent with the 2D profile of the imaged object. The model comes to equilibrium as soon as its 3D shape achieves maximal consistency with the image data.

reconstruction scenario in the monocular case.

The image potential is generalizable to exploit more extensive profile information in multiple images acquired from different viewpoints around an object. For the particular case of stereo, the potential incorporates two images from slightly different vantage points. Deprojection of its gradient through a binocular camera model creates a stereo force field in space. Points on the model's occluding boundaries with respect to both the left and right eye are sensitive to the stereo force field. The forces position boundary points laterally and in depth such that their binocular projections coincide as much as possible with object profiles in both images.

The ambient force field becomes dynamic when imaged objects are in motion. It carries the model through nonrigid motions, continually molding its shape to maintain maximal consistency with the evolving image data. The evolution of the model is computed by numerically integrating the partial differential equations of motion for the deformable body as it reacts to the dynamic force field.

## A. The Symmetry-Seeking Model and Intrinsic Forces

The "construction" of a symmetry-seeking model is straightforward. Take a deformable sheet made of elastic material (a membrane–thin-plate hybrid). Roll this sheet to form a tube. Next, pass a deformable spine made of similar material down the length of the tube. Couple the spine to the tube with radially projecting forces that induce axisymmetry. The rigidities of the spine and the tube are independently controllable, and their natural rest metrics and curvatures can either be prescribed or modified dynamically. If the circumferential metric of the tube is set to zero, for instance, the tube will tend to contract around the spine, unless the other forces prevail. The model will shorten or lengthen as the longitudinal metrics of the tube and spine are modified. In short, a variety of interesting behavior (including viscoelasticity and fracture) can be obtained by adjusting the control variables designed into the symmetry-seeking model.

We represent the spine and tube as geometric mappings from material coordinate domains into Euclidean 3-space. The spine is a deformable space curve, while the tube is a deformable space sheet. Functionals characterize the deformable materials by associating non-negative strain energies with the mappings. The strain energies of

the deformable spine and tube are expressed by second-order controlled-continuity spline functionals.[39] These functionals contain weighting functions which vary over the material coordinate domains. One set of weighting functions controls the local tension of the deformable material, while a second set controls its local rigidity. In addition to controlling elastic properties, along with the natural metrics and curvatures of the material at rest under zero strain, the weighting functions can be set to zero at any point to permit position or tangent discontinuities to occur in the material (hence the name controlled-continuity spline).

In addition to the elastic forces due to the deformable material, which are given by a variational derivative of the strain energy functionals, the intrinsic forces of the model include three symmetry-seeking forces that couple together the spine and tube components. The first force coerces the spine into an axial position within the tube. The second force encourages the tube to be radially symmetric around the spine. The final force provides control over expansion and contraction of the tube around the spine. The strengths of each of these forces is adjustable over the length of the spine. For example, the tube will have a tendency to inflate wherever the strength function of the third force is positive or deflate whenever it is negative. In particular, this force can be used to cinch shut the two ends of the tube by assigning large negative values to the strength function at $s = 0$ and $s = 1$.

The nonrigid motion of the spine and tube are dictated by continuum mechanical equations for deformable bodies under applied forces. Hence, the symmetry-seeking model is governed by a pair of partial differential equations (one for $\mathbf{v}^S$ and one for $\mathbf{v}^T$) coupled through the symmetry-seeking forces. The (hyperbolic-parabolic) system is second-order in time and fourth-order in the material domains. The equations dictate that the net extrinsic force is balanced by the inertial forces due to the mass density of the deformable body, plus the damping forces due to dissipation, plus the intrinsic forces which include elastic forces and symmetry-seeking forces. The extrinsic forces are expressed as variational derivatives of external potential functions which are described in the next section.

The continuous differential equations of motion for the symmetry-seeking model pose a nonlinear initial-boundary-value problem. To obtain a numerical solution, we first perform a semi-discretization in the material coordinates of the model using standard finite-difference methods. Our reconstruction algorithms integrate the resulting coupled system of second-order ordinary differential equations through time using a semi-implicit Euler time-integration scheme. Numerically, this amounts to solving a sequence of dynamic equilibrium problems, each solution providing initial conditions to the subsequent problem. An operator splitting approach, as used in alternating direction implicit (ADI) methods, has been efficacious on our rectangular computational grids. Splitting the controlled-continuity spline operator yields unidimensional systems with pentadiagonal coefficient matrices. These systems are efficiently solvable (linear-time in the number of unknowns) using direct solution methods. We employ a normalized Cholesky decomposition step followed by a forward-reverse resolution step. The direct solution of each unidimensional system obtained through operator splitting "immediately" distributes to all nodes along two perpendicular parametric grid lines the effects of forces acting on their common node.

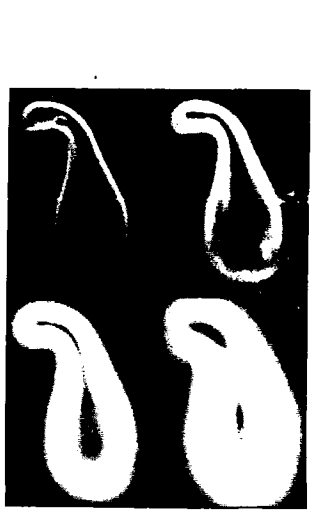


Figure 14: Result of applying image operations to the squash image of Fig. 2. Smoothing $\sigma$ increases progressively from left to right and top to bottom. Darkness indicates magnitude of local gradient of the Gaussian blurred image. Each image has been rescaled to span the available intensity range.

## B. Extrinsic Force Fields

We transform input images into generalized potential functions suitable for reconstruction. The force fields resulting from these potentials bring symmetry-seeking models into maximal consistency with the images, and maintain the consistency over time in the dynamic case. More specifically, profiles in the images exert an attraction over the model such that the deformable tube, as projected into the image planes through a suitable camera transformation, accounts as much as possible for the observed profiles.

To simplify the potential functions, we consider objects with subdued texture which are imaged in front of a contrasting background. Hence, we can expect that the stronger image intensity gradients are associated with object profiles. To define a potential we first compute the magnitude of the gradient of the image convolved with a Gaussian smoothing filter of width $\sigma$. Figure 14 illustrates the effect of these image operations for progressively larger values of $\sigma$.

The ambient force field stemming from this potential attracts the occluding boundary (with respect to the imaging viewpoint) of the deformable tube towards significant image intensity gradients. The 3D shape of the model's occluding boundary in space aligns to maximize the magni-

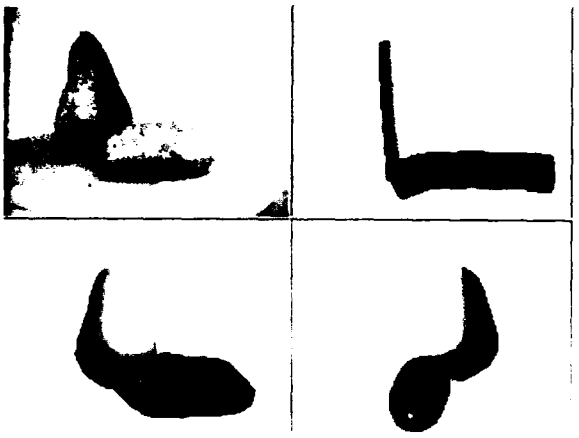


Figure 15: Reconstruction of a still-life scene with symmetry-seeking deformable models. (upper left) Image of the scene. (upper right) Initial, user-specified configurations of the 3D models. Two graphically rendered views of the reconstructed still life: (lower left) Frontal view of the 3D models; (lower right) side view.

tude of the image gradient (darkness) along the projected model profile in the image. The force field thereby deforms the symmetry-seeking model to make its shape consistent with the object's profile in the image, with the remaining surface following smoothly.

By creating potential functions at several scales, we can trade-off localization accuracy against long range attraction. As is evident from figure 14, broad wells surround the local minima of the image potential at the coarser scales. Such wells attract the model from a considerable distance, but the associated minima are blurred and localize the profiles in the image data rather poorly. Continuous scale-space provides a good medium for obtaining both long range attraction and good localization. We can apply a continuation method in the image potential scale space, parameterized by $\sigma$, which allows the model to equilibrate at a coarse scale, then continuously reduces the smoothing $\sigma$ to track an equilibrium trajectory from coarse to fine scale [46].

Figure 15 illustrates the reconstruction of two quasi-symmetric objects, a pear and a potato, from a single image. The grey-level image of the still life scene is shown on the upper left. Notice that the potato partially occludes the pear in the image. The initial model configurations manually specified by the user are shown on the upper right. The lower part of the figures shows the reconstructed 3D models from two points of view. To handle the partial occlusions (incomplete boundaries), we must desensitize occluded parts of the model from image forces. We employ a standard 3D ray casting technique in conjunction with the imaging projection in order to test surface patches for visibility using a depth buffer. The ray casting operation requires knowledge of the relative depth ordering of the objects. In the monocular case, depth information is not available directly, so the user presently specifies the depth ordering.

We can obtain true depth information through stereo, however. In the binocular case, the deformable model is matched to an object's profiles in a stereo image pair. We make the occluding boundaries with respect to both left and right image viewpoints sensitive to the ambient force field. The symmetry-seeking model is free to undergo motion in 3-space while deforming, such that its stereoscopic projection though the binocular camera model best accounts for the observed profiles in both images. The image processing operations are the same as for the monocular case.

Profiles of smooth objects and the occluding boundaries which generate them are known to present difficulties to conventional stereo matching techniques. This is mainly because profiles observed in the left and right image map to different occluding boundaries on smooth objects. Our method for reconstructing models directly from images overcomes this problem. The use of a full 3D model in tandem with separate left and right projection operators simplifies the association of each image profile with the corresponding occluding contour on the model.

Figure 16 shows the binocular reconstruction method applied to a stereo image of a human finger. The user specifies an initial spine and the initial tube is a cylinder around the spine. The model's differential equations are solved to reconstruct the shape of the object in proper depth. The figure shows the reconstructed shape rendered from several viewpoints.

When the input images are time-varying the ambient force field becomes dynamic. It carries the model through motions, continually molding its shape to maintain maximal consistency with the evolving image data. Our method for allowing the symmetry-seeking model to track an object undergoing nonrigid motion is as follows. The first frame of the image sequence is presented to the model as if it were a static scene. The model achieves the best possible reconstruction using this initial data. The projected boundary points equilibrate at a fixed point in the ambient force field and the model locks on to the consistent state.

Whereas the equilibrium persists indefinitely in the static case, in the dynamic case we immediately present the model with the next frame of the image sequence. Now the ambient force field is perturbed due to the motions of objects in the scene. The model actively seeks a new consistent state by moving towards the nearest fixed point. If the motion of the object is sufficiently slow and continuous, the model will track the dynamic equilibrium point, thus updating its state in accordance with the new image information available to it. By repeating this procedure with each successive incoming frame, the symmetry-seeking model integrates the incoming information over time.

As an illustration, we use a stereo-motion sequence consisting of 40 video fields that portray the 3D motion of the human finger. We use as initial condition the equilib-

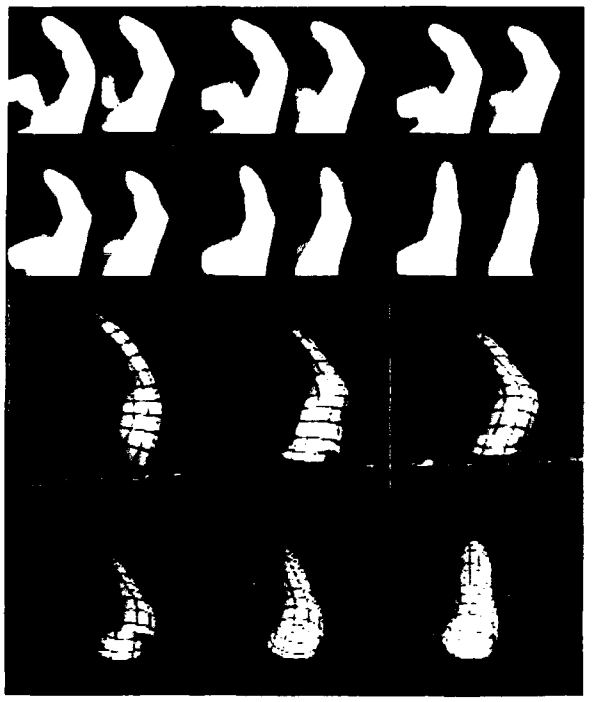


Figure 16: Evolution of the reconstructed 3D finger model through time. (top) Six frames of the stereo motion sequence. (bottom) Evolving shape and motion of the model.

rium shape computed on the first frame of this sequence. The equations of motion are then integrated through time over the remaining frames of the stereo sequence. This produces a dynamic 3D reconstruction of the finger's non-rigid shape and motion. Figure 16 shows six representative frames of the sequence along with the corresponding reconstructed shapes.

In the restricted case of computing 3D models of objects from a single monocular image, we tacitly assume a suitable viewpoint wherefrom all significant object features are visible, and we assume that the axis of the object is not severely inclined away from the image plane. However, in the more general case involving stereo information, we experience little difficulty in tracking the shape and motion of an object, even if it's axis tilts away from the image plane significantly.

## C. Discussion

A number of features distinguish our modeling approach from the norm in computational vision. Conventional models of 3D shape are purely geometric, hence passive. By contrast, our deformable models are active. They react to extrinsic forces as one would expect real elastic objects to react to applied forces. This is because deformable models are dynamic models governed by physics; specifically, by the principles of elasticity theory as expressed through Lagrangian dynamics.

The *distributed* nature of deformable models enhances their representational power. Every material point potentially contributes three spatial degrees of freedom which are mutually constrained with variable tightness by the model's intrinsic forces. Shape representations capable of applying constraint in a controlled manner are desirable for reconstruction and recognition. An immediate advantage is that the geometric coverage of deformable models can be significantly broader than *lumped*-parameter families of shapes such as the superquadric models that have applied to computer vision.[33] Lumped-parameter models, while relatively inexpensive to work with, are capable of accurately representing only a restricted class of artificial objects. This is because they "wire into the parameterization" a relatively small family of shapes, rather than place generic constraints on shape as do our deformable models. Lumped-parameter models cannot immediately accommodate most natural objects of interest, so precise hierarchical subdivision and parameterized deformations become practical necessities to contend with. By contrast, the free-form flexibility of our deformable models renders them immediately adaptable to natural shapes.

The force fields that we employ yield interesting results, their simplicity notwithstanding. However, the main shortcoming of our current algorithms stems from the fact that profile information alone, even moving stereo profiles, provide incomplete information about objects. When no image data is directly available over substantial portions of the object's surface, the symmetry-seeking material may yield an inaccurate reconstructed shape. Also, it is difficult to detect rotations around the object's axis exclusively from moving profile information. Region-based measures over the surface would be helpful in this regard.

Despite the limitations of our current implementation, a crucial advantage of our approach is the ease of integrating additional constraints into the solution. For instance, we can straightforwardly generalize the binocular potential function to integrate any number of views taken from known viewpoints around the object. We can formulate extrinsic constraints that exploit shading and texture information over the entire visible surface. Stereo-motion constraints based on local area correlation promise to effectively supplement our current edge-based information. By applying more sophisticated image processing methods, we expect to obtain extrinsic forces that can deal with textured objects and more general imaging conditions.

# IV. Spacetime Constraints[3]

[3]The material in this section is adapted from the paper "Spacetime Constraints" by Andrew Witkin and Michael Kass, submitted

Computer animation has made enormous strides in the past several years. In particular, Pixar's *Luxo, Jr.*[4] [34] marked a turning point as perhaps the first computer-generated work to compete seriously with works of traditional animation on every front. Key among the reasons for *Luxo, Jr.'s* success is that it was made by a talented animator who adapted the principles of traditional animation to the computer medium. *Luxo, Jr.*, in large measure, *is* a work of traditional animation that happens to use a computer to render and to interpolate between keyframes. John Lasseter spelled this out clearly in his presentation to SIGGRAPH '87 [26].

Although *Luxo, Jr.* showed us that the team of animator, keyframe system, and renderer can be a powerful one, the responsibility for defining the motion rests almost entirely with the animator, who must provide enough keyframes too ensure good splining. If spline interpolation could be replaced by something that knows more about the motion being performed, the division of labor could be improved.

Some aspects of animation—personality and appeal, for example—will surely be left to the animator's artistry and skill for a long time to come. However, a reading of Lasseter's paper reveals that many of the principles of animation are concerned with making the character's motion look *real* at a basic mechanical level that ought to admit to formal physical treatment. Consider for example anticipation, squash and stretch, and follow-through as elements of a jump. Any creature—human or lamp—can only accelerate its own center of mass by pushing on something else. In jumping, the opportunity to control acceleration only exists during contact with the floor, because while airborne there is nothing to push on. Anticipation before the jump is the phase in which the needed momentum is acquired, by squashing then stretching to push off against the floor, and follow-through is the phase in which the momentum on landing is absorbed.

Such physical arguments make nice *post hoc* explanations, but can physics be brought to bear in *creating* the complex active motions of characters like Luxo? If so, how much of what we regard as "nice" motion can be derived from first principles, and how much is really a matter of style and convention?

This section presents the initial results of our effort to answer these questions. Our specific objective was to obtain realistic and pleasing motion, exhibiting as many as possible of the principles of animation by specifying:

- *What* the character has to do, for instance "jump from here to there."
- *How* the motion should be performed, for instance "don't waste energy," or "come down hard enough to splatter whatever you land on."
- What the character's *physical structure* is—what the pieces are shaped like, what they weigh, how they're connected, etc.
- What physical *resources* are available to the character to accomplish the desired motion, for instance the character's muscles (or whatever an animate lamp has in place of muscles,) a floor to push off from, etc.

Our chosen object of study was a Luxo model of our own design, and the basic task we set it was to execute a convincing jump, telling it only where to start and where to end. Before describing our methods and results, we begin by reviewing current uses of physical methods for animation.

## A. Background and Motivation

Recently, there has been considerable interest in incorporating physics into animation using simulation methods. [21, 53, 3, 41, 16, 19] The appeal of physical simulation as an animation technique lies in its promise to produce realistic motion automatically by applying the same physical laws that govern real objects, rather than relying on the animator's skill at keyframing to create convincingly physical motion.

Unfortunately, the realism of simulation-based animation comes at the expense of control. The course of a simulation is completely determined by the objects' initial positions and velocities, and by the forces that act on the objects along the way. Consequently, the animator's sole *means of controlling* what happens is by choosing initial conditions and by injecting forces into the system.

The animator who wants to use bare physical simulation to create anything more coordinated than chaotic tumbling must therefore come up with a combination of initial conditions and applied forces that achieve the desired motion. Anyone who has actually tried this knows that even in the simplest cases it is an excruciatingly painful matter of trial-and-error. For instance, while it is easy enough to simulate the behavior of a bouncing ball, making the ball bounce *to* a particular place requires finding just the right starting values for position, velocity, and spin. Using simulation methods, one generally makes a guess, runs the simulation to see where the ball ends up, and starts again with a new and hopefully better guess. As the situation grows more complicated, this haphazard approach quickly becomes hopeless. To control an animate character this way, it would be necessary to specify all the muscle forces at each instant in time. This would be like trying to control a robot arm by manually varying the motor torques.

Lack of control is the basic problem limiting the usefulness of simulation as an animation technique. Ideally, one would like a method that combines the advantages of keyframing with those of simulation, providing the animator with the ability to directly specify or constrain any aspects of the motion—where things start, where they wind up, and how they get there—and allowing those aspects of motion that follow directly from physics to emerge auto-

to SIGGRAPH '88

[4] "Luxo" is a trademark of Jac Jacobsen Industries AS.

matically within the framework imposed by the animator. Evidently, simulation is far from this ideal.

In an effort to reconcile the advantages of simulation with the need for control, several researchers [3, 21] have proposed methods for blending positional constraints with dynamic simulations. The idea behind these methods is to treat kinematic constraints as the consequences of unknown "constraint forces," solve for the forces, then add them into the simulation, exactly canceling that component of the applied forces that fights against the constraints. In principle, constraint forces are equivalent in their effect on the system to enormously stiff springs that keep pieces firmly attached in the face of any applied force, although the methods used to compute them avoid the practical difficulties that actual stiff springs would introduce.

Constraint forces offer a good way to model mechanically coupled objects, particularly jointed or articulated bodies. However, as a means of controlling animation they offer only the ability to drag objects around with giant invisible springs, which fails to solve the basic control problem posed above. Consider again the example of the bouncing ball whose desired destination is known. Were we to use constraint forces to drag the ball straight to its destination, we would see in the resulting motion a ball being pulled by an invisible hand. To make the motion look natural, it would be necessary to drag the ball along just the right physical-looking trajectory, reducing the entire physical simulation to nothing more than a roundabout and very inefficient way to do keyframing.

We want the ball to arrive at its appointed destination "under its own steam" influenced by the force of the hand that threw it, by gravity, by collisions with other objects, and so forth—in short, by legitimate forces, not by the action of spurious springs. At the same time, we want it to end up where we want it to end up. To accomplish this, we need constraints that are more like stage directions than springs, doing their work without introducing stray forces. Moreover, their effects must propagate backward as well as forward through time, influencing, for instance, the way the ball was thrown at the start of the motion to bring it to its target at the end.

These requirements led us to a new formulation of the constraint problem, whose central characteristic is that we solve for the motion of the system over the entire time interval of interest, rather than progressing sequentially through time. Because we extend the model through time as well as space, we call the formulation *spacetime constraints*.

In an ordinary simulation, the *state* of the system—the values of all the variables of position and velocity, together with the values of other independent variables—is maintained at the current instant in simulation time, and updated as the simulation proceeds. Each of these state variables is a function of time, but only the values at the current instant need be remembered. In a spacetime system, these functions of time are represented explicitly, from beginning to end. Each function might be represented as a spline or simply a sequence of values. Similar functions are maintained for time-dependent forces, such as those generated by muscles. The state of the spacetime system is the set of values of all the variables used to control the time functions. If a spline representation is chosen, the state consists of the knot points; if a sequence of values is chosen, the state consists simply of the values themselves.

Encapsulated in the state of the spacetime system is every modeled aspect of the physical system's behavior over the time interval of interest—position, velocity, acceleration, kinetic energy, force, etc. Some states of the spacetime system denote physically valid motions, while others do not. We can express physics as a *constraint* on the system whose basic form is $f - ma = 0$. If we read off the forces and accelerations and discover that this relation holds at all times, the state represents a physically valid motion, otherwise it does not.

The idea of spacetime constraints is simply to combine this "physics constraint" with kinematic constraints that specify positions, velocities, etc., of various objects at various times, and solve for a state that satisfies both sets of constraints. Additionally, where this solution is underdetermined, we may add criteria that say *how* the motion is to be performed within the kinematic and physics constraints. These last take the form of functions to minimize, and may be used, for instance, to achieve optimally smooth or efficient motion. Solving the spacetime system reduces to a standard problem of constrained optimization.

The spacetime formulation provides the sought-after combination of physics and control. The kinematic constraints have precisely the form—and the function—of keyframes in conventional animation, but we interpolate with physics instead of cubics.

The remainder of the paper is organized as follows: in the next section we illustrate the spacetime formulation using a toy example moving particle. Then, we outline the spacetime model for an animate Luxo lamp. Finally we present results obtained using the Luxo model.

## B. A spacetime particle

Spacetime methods are complicated, more so than ordinary physical simulations. Instead of starting with initial conditions and solving sequentially through time, we compute the entire path in a single constrained optimization.

As a gentle but concrete introduction, we begin with a minimal example: a moving particle, influenced by gravity, and possessed of a "jet engine" as a means of locomotion. With no restrictions on its jet forces, the particle can move any way it likes. The problem we formulate here is that of obtaining the particle's motion and jet forces as a function of time, such that (a) physics holds, (b) some positional constraints are satisfied, and (c) the consumption of "jet fuel" is minimized. Here, (a) and (b) are the constraints and (c) provides the function to be optimized. Although too simple to produce any really interesting motion, this problem exhibits all the key elements of the method, and

will aid in understanding what follows.

**Formulating the problem.** Problems in constrained optimization are ordinarily expressed by specifying a state vector of moving variables, a collection of constraint functions and an objective function to be minimized. The constraint functions are functions $v_i$ that go to zero when the constraints are satisfied. For spacetime systems, the state vector specifies the entire motion over a particular time interval.

There exists an extensive literature treating the numerical solution to problems of this form, a review of which is well beyond the scope of this paper. For practical treatments of the subject, as well as more specialized references, we recommend [37] and [15].

In order to cast the moving particle in the framework of constrained optimization, we begin by choosing the moving variables. Clearly, we need to represent the position $\mathbf{x}(t)$ of the particle over time. In addition, we want to compute the changing thrust $\mathbf{j}(t)$ of the particle's jet engine. There are two basic choices for representing these functions of time. The first is to use a set of discrete time samples of the functions and approximate their derivatives with finite differences. The second method is to represent the functions by coefficients of local basis functions (finite elements). We use the simpler finite difference approach here.

Using the finite difference approach we represent $\mathbf{x}(t)$ and $\mathbf{j}(t)$ by vectors of time samples. The time derivatives $\mathbf{v} = \partial\mathbf{x}/\partial t$ and $\mathbf{a} = \partial^2\mathbf{x}/\partial t^2$ are obtained from the $\mathbf{x}$ array according to the finite-difference formulas $\mathbf{v}_i = (\mathbf{x}_i - \mathbf{x}_{i-1})/h$, and $\mathbf{a}_i = (\mathbf{x}_{i+1} - 2\mathbf{x}_i + \mathbf{x}_{i-1})/h^2$, where $h$ is the time-difference between samples. The *state* of the spacetime system consists of the set of values in the two vectors.

Given these moving variables, we want to express the constraint that the motion is physically valid. This constraint is given by Newton's second law. In order to ensure that $\mathbf{f} = m\mathbf{a}$, we construct a constraint function equal to the difference $\mathbf{f} - m\mathbf{a}$. When the constraint function vanishes, we have physically valid motion. The total force $\mathbf{f}$ on the particle is the sum of the jet thrust $\mathbf{j}$ and the gravitational force $m\mathbf{g}$, where $m$ is the mass of the particle and $\mathbf{g}$ is the constant of gravitational acceleration. To keep the example simple, we will assume that the mass of the particle is much greater than the mass of the fuel, so $m$ can be regarded as a constant over time. Then the quantity we want to constrain to be zero is just $\mathbf{V} = m\mathbf{g} + \mathbf{j}(t) - m\mathbf{a}$. The constraint should be valid at all time samples, so if we insert the finite difference formulas, and write $\mathbf{V}$ in terms of the state variables $\mathbf{x}_i$ and $\mathbf{j}_i$, we obtain the constraint functions

$$\mathbf{v}_i = \mathbf{f}_i - m\mathbf{a}_i = \mathbf{j}_i + \mathbf{g} - \frac{m}{h^2}(\mathbf{x}_{i+1} - 2\mathbf{x}_i + \mathbf{x}_{i-1}),$$

which express the *physical compatibility* of the motion and the forces.[5]

The power of the spacetime formulation comes when the physics constraint is combined with kinematic constraints that specify what the motion is supposed to do: a simultaneous solution to both yields a motion that does what we want in a physically valid way. For instance, to specify where $P$ must be at certain times, we may add constraints of the form $\mathbf{y}_a - \mathbf{x}_a = 0$, where $\mathbf{y}_a$ is the desired position at time $t_a$. Similar constraints can be constructed to specify velocity, to restrict the particle to a curve or surface, etc.

Even with the addition of a number of kinematic constraints, the solution is generally underdetermined, because $\mathbf{j}$ allows us to exert an arbitrary force on $P$ at each instant in time. This state of affairs is typical of animate models. Within the hard kinematic and physical constraints, we may therefore provide criteria that specify *how* the motion should be performed, expressed as functions that say how "good" a particular valid force-and-motion combination is compared to others. Then of all the states satisfying both the physical and kinematic constraints we may seek the "best" one by the specified criteria. For instance, supposing we are interested in fuel economy, we might try to minimize $\sum_i |\mathbf{j}_i|^2$, summing over time samples, on the assumption that burn rate is proportional to thrust. More generally, different choices of the optimization function may be used to impose criteria of efficiency, smoothness, or style.

**Solving the problem.** We are now equipped to pose problems such as "move the particle from point $\mathbf{x}_0$ at time $t_0$ to a point $\mathbf{x}_n$ at time $t_n$ minimizing fuel consumption," with ready extension to more elaborate kinematic constraints. This particular problem can be written as the following constrained optimization

$$\begin{aligned} &\text{Minimize} && \sum_{i=0,n} |\mathbf{j}_i|^2 \\ &\text{Subject to} && \\ \mathbf{j}_i + \mathbf{g} - \frac{m}{h^2}(\mathbf{x}_{i+1} - 2\mathbf{x}_i + \mathbf{x}_{i-1}) &= 0, && 0 \le i < n \\ \mathbf{x}_0 - \mathbf{a} &= 0 && \\ \mathbf{x}_n - \mathbf{b} &= 0 && \end{aligned}$$

To solve this problem, we use a modified iterative Newton-Raphson method [37]. Let $v_i$ be a single vector containing all the constraint functions and let $q_j$ be a single vector containing all the state variables. We begin each

[5] It is worth mentioning briefly that $f - ma = 0$ is not the only form that the physics constraint can assume. A particularly attractive alternative is *Hamilton's Variational Principle* [18] which states that the motion between any two fixed states always extremizes the integral $\int_{t0}^{t1} T - V dt$, where $T$ and $V$ are kinetic and potential energy respectively. A shortcoming of Hamilton's principle, which led us to reject it as a constraint, is its requirement of fixed boundary conditions, i.e. full keyframes at the beginning and end, which is unnecessarily restrictive. A minor problem with Hamilton's principle is that it is awkward to incorporate non-conservative forces.

iteration by solving

$$-h_i = \frac{\partial h_i}{\partial q_j}\hat{q}_j \tag{9}$$

for a step $\hat{q}_j$ that would drive the objective functions $h_i$ to zero without regard to the constraints. Equation 9 is a linear system which we solve using a conjugate gradient algorithm from [37] to compute the pseudo-inverse solution [25, 16] while exploiting sparsity. Due to the least squares norm of the pseudo-inverse, solving equation 9 has the effect of minimizing $\sum h_i^2$. We next compute another update $\tilde{q}_j$ by solving

$$-v_i = \frac{\partial v_i}{\partial q_j}(\hat{q}_j + \tilde{q}_j) \tag{10}$$

where the Jacobian $\partial v_i/\partial q_j$ is evaluated at $q_j$. Equation 10 is another linear system solved in the same way. The update $\tilde{q}_j$ simultaneously drives the constraint functions $v_i$ to zero and cancels any component of $\hat{q}_j$ which to first order would increase the magnitude of $v$. Finally, we make the update $q_j \leftarrow q_j + \hat{q}_j + \tilde{q}_j$. This algorithm is related to the one used by Barzel & Barr [3] to compute constraint forces.

If the constraints are locally linear, then the solution to equation 10 will drive them all simultaneously to zero. If the constraints are non-linear, then the procedure must be used iteratively. We have found the algorithm to be very robust and speedy—some of the Luxo examples to be shown later converged in four steps.

It is critical to make use of sparsity in $\partial v_i/\partial v$ in order to obtain acceptable performance for non-trivial constraint systems. The particle example yields a typical sparsity pattern:

$$\begin{aligned}
\frac{\partial \mathbf{F}_i}{\partial \mathbf{x}_j} &= 2m/h^2, \quad i = j \\
&= -m/h^2, \quad i = j \pm 1 \\
&= 0, \quad \text{otherwise} \\
\frac{\partial \mathbf{F}_i}{\partial \mathbf{j}_j} &= 1, \quad i = j \\
&= 0, \quad \text{otherwise},
\end{aligned}$$

where

$$\mathbf{F}_i = \mathbf{j}_i + \mathbf{g} - \frac{m}{h^2}(\mathbf{x}_{i+1} - 2\mathbf{x}_i + \mathbf{x}_{i-1}).$$

It should be clear that most of the entries in $\partial v_i/\partial v$ are zero for this case.

We conclude the particle example with an observation about the role of such forces as $\mathbf{j}(t)$. Although these time-dependent forces are computed as part of the solution process, forces can't be *seen*, at least not directly, and play no role in the rendering process. From the standpoint of animation, the computed forces are necessary byproducts of the creation of realistic motion. In principle, at least, the problem we solve is closely related to the one of controlling a real mechanism, for instance a robot. For that problem, it is the motion that is a byproduct and the forces that are used directly to drive the motors. Although many of the complexities of robot control—motor inertia, backlash, cable stretch, etc.—may be safely ignored in animation, the connection is an interesting one. Path optimization methods have been used in robotics (see [7] and [8] for recent surveys,) but the central concern has been with the very different problem of path planning for obstacle avoidance.

## C. Spacetime Luxo

We are now equipped to proceed to a more interesting spacetime model, specifically a model of an animate Luxo Lamp. In principle, the Luxo model has the same form as that of the particle described in the last section. In practice things are far more complicated because the equations of motion are more complex with more degrees of freedom. The differentiation of the physics constraint with respect to state variables produces a great many large and unattractive expressions. The number and complexity of these derivatives is large enough that deriving and coding them by hand would be all but hopeless. Faced with this bleak prospect, the authors implemented a lisp-based system for symbolic differentiation, simplification, and optimized code-generation of tensor forms. This tool relieved us of the task of cranking out the derivatives, automatically producing some 4000 lines of code to evaluate the large, sparse derivative matrices. Space does not permit reproducing the derivative forms here, nor would it be particularly enlightening to do so. Rather, our goal in this section is to describe the Luxo model at the level that we were required to specify by hand, as input to our math compiler. Those who wish to duplicate our results without implementing a symbolic math package of their own face two choices: adapt an available symbolic math system, such as Macsyma, to the task, or perform the differentiations numerically at runtime.

Geometrically, our simplified Luxo is composed of rigid bodies of uniform mass connected by frictionless joints. Each joint is equipped with a "muscle" modeled as an angular spring whose stiffness and rest angle are free to vary with time. Our lamp is subject to the forces of its own muscles, in addition to the external force of gravity and the contact forces arising from its interaction with objects such as floors and skijumps. In the treatment and examples to follow, we restrict Luxo's motion to a plane. This expedient greatly simplifies the mathematics, while still allowing the creation of complex, subtle, and interesting motion. A picture of the model appears in figure 17.

Newton's second law suffices to write down the equations of motion for arbitrarily complex mechanical systems. However, deriving equations of motion for complex constrained systems directly from $\mathbf{F} = m\mathbf{a}$ can be a laborious and error-prone task. An alternative and equivalent formulation of mechanics, due to Lagrange, sometimes allows

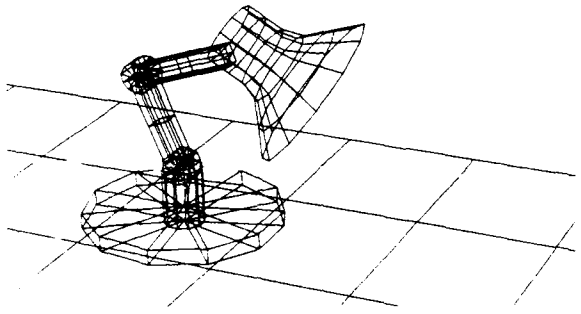


Figure 17: Luxo

for a more direct derivation of the equations of motion. The Lagrangian equations of motion can be written [18]

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial T}{\partial \mathbf{q}} - \mathbf{Q} = 0, \tag{11}$$

where $T$ is kinetic energy, $\mathbf{q}$ is the vector of state variables, $\dot{\mathbf{q}}$ is their time derivative, and $\mathbf{Q}$ is the *generalized force* on $\mathbf{q}$. A generalized force is just a real-world force mapped into the object's arbitrary state space. In the case of a conservative force expressed as a scalar potential $V(\mathbf{q})$, the generalized force is just $-\nabla V$. In the case of a force applied to a point $\mathbf{x}$ on the object, $\mathbf{Q} = \mathbf{x}\partial\mathbf{x}/\partial\mathbf{q}$.

Constructing the physics constraint using Lagrange's equation involves the following steps:

- Construct an expression for the generalized force, $\mathbf{Q}$.
- Express the kinetic energy, $T$, of the body in terms of the position variables and their first time derivatives.
- Perform the differentiations indicated in equation 11, to obtain Lagrange's equation of motion.
- Re-express Lagrange's equation in terms of spacetime state variables, replacing time derivatives by finite differences to obtain the physics constraint.
- Differentiate the resulting expression with respect to all state variables to obtain the jacobian of the physics constraint.

The first three steps create the equations of motion needed to perform an ordinary simulation, producing an expression analogous in form to $\mathbf{f} - m\mathbf{a} = 0$. The remaining ones are peculiar to the spacetime formulation. Only the first two steps will be described here.

Computing the generalized force is not usually difficult. For conservative forces, which may be expressed as gradients of scalar potentials, $\mathbf{Q}$ is obtained by writing the potential function in terms of the object's state variables, then taking a gradient. For non-conservative forces that are expressed directly in terms of state variables, $\mathbf{Q}$ is just the force itself. Forces given in terms of quantities that depend on the state variables, such as the position of a point on the body's surface, must be multiplied by the jacobian of those quantities with respect to the state variables to obtain $\mathbf{Q}$. Specifics will be given when the forces are discussed below.

The kinetic energy, $T$, of any body is the integral over the body of the kinetic energy of each particle, $\frac{1}{2}\rho\,|\dot{\mathbf{x}}|^2$, where $\rho$ is the mass density at point $\boldsymbol{x}$. The kinetic energy of an articulated object is the sum of the kinetic energies of the parts. Each of Luxo's links is modeled as a rigid body rotating about an axis of fixed direction that passes through the origin in body coordinates (see figure 18.) Because the axis is fixed, the orientation of the $i$-th link may be denoted by a single angle $\theta_i$, with angular velocity $\omega_i = \theta_i\mathbf{a}$, where $\mathbf{a}$ is a unit vector in the direction of the axis. In addition to rotation, the body origin undergoes a translation $\mathbf{p}_i$, with translational velocity $\mathbf{v}_i = d\mathbf{p}_i/dt$. Each link has mass $m_i$, a constant moment of inertia $I_i$ about the rotation axis, and a center of mass $\mathbf{c}_i$ expressed as a displacement from the body origin. In these terms, the kinetic energy of the $i$-th link is

$$T_i = \frac{1}{2}m_i\,|\mathbf{v}_i|^2 + m_i\omega_i \cdot \mathbf{v}_i \times \mathbf{c}_i + \frac{1}{2}\,|\omega_i|^2\,I_i. \tag{12}$$

To connect the links, each link inherits as its translation the position of the previous link's endpoint, with the base's translation, $\mathbf{P}$, serving as a translation parameter for the whole model. The translational velocity $\mathbf{v}_i$ of the $i$-th link is thus

$$\begin{aligned} \mathbf{v}_i &= \frac{d\mathbf{P}}{dt}, \quad i = 0 \\ &= \mathbf{v}_{i-1} + \mathbf{r}_{i-1} \times \omega_{i-1}, \quad \text{otherwise} \end{aligned}$$

where $\mathbf{r}_{i-1}$ is a vector from the $(i-1)$-th link's center of rotation to its point of attachment with the $i$-th link. The total kinetic energy $T$ is obtained by recursively substituting this expression into equation 12 to obtain the $T_i$'s, and summing over $i$.

To obtain Lagrange's equation, the kinetic energy derivatives to be computed are

$$\frac{\partial T^2}{\partial \dot{\mathbf{P}}\partial t}, \quad \frac{\partial T^2}{\partial \omega^i \partial t}, \quad \frac{\partial T}{\partial \theta^i},$$

where $\dot{\mathbf{P}}$ is the translational velocity of the base, and the $\omega^i$'s and $\theta^i$'s are respectively the angular velocities and orientations of the four links. To obtain the kinetic energy terms of the physics constraint's jacobian, these derivatives must again be differentiated with respect to the vector of spacetime state variables, after finite differences have been substituted for time derivatives.

Luxo's muscles are three angular springs, one situated at each joint. The spring force on the joint connecting the $i$-th and $(i+1)$-th links is defined by

$$F_i = k_i(\phi_i - \rho_i),$$

where $k_i$ is the stiffness constant, $\phi_i$ is the joint angle, and $\rho_i$ is the rest angle. Our model is parameterized by link

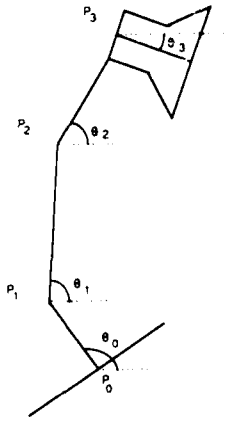



Figure 18: Each link of the Luxo model rotates about the previous link, inheriting the previous link's end position as a translation.

orientations rather than joint angles. The joint angle is $\phi_i = \theta_{i+1} - \theta_i$, the difference between the orientations of the surrounding links. The generalized force on $\theta_i$, the orientation of the $i$-th link, due to the $j$-th muscle is

$$\begin{aligned} Q_i &= F_j \frac{d\phi_i}{d\theta_j}, \\ &= k_j(\phi_j - \rho_j), \quad j = i+1 \\ &= -k_j(\phi_j - \rho_j), \quad j = i \\ &= 0, \quad \text{otherwise} \end{aligned}$$

Unlike passive springs whose stiffness and rest state are constants, $k_i$ and $\rho_i$ vary freely over time, allowing arbitrary time-dependent joint forces to be exerted. The imposition of active forces by varying stiffness and rest state is actually a fair approximation to the way real muscles work. As with all the other independent variables in a spacetime system $k_i$ and $\rho_i$ for each of the three muscles are represented by spacetime arrays, whose contents are moving variables to be solved for.

## D. Results

**Jumping Luxo.** To create jumping motion, we used kinematic constraints to specify initial and final poses, using linear interpolation between them to create a trivial initial condition for the spacetime iteration. Another constraint was used to put Luxo on the floor during the initial and final phases of the motion. Subject to these and the physics constraint, we minimized the power due to the muscles, $F_\theta \dot{\theta}$. In one variation, we adjusted the mass of Luxo's base, leaving the situation otherwise unchanged. In another, we additionally constrained the force of contact with the floor on landing, to produce a relatively soft landing. In a final variation, we added a hurdle, together with a constraint that the jump clear the hurdle.

The pose constraints consisted of values for the three joint angles, and were applied to the first two and last two frames of motion. Because we measure velocity using a finite difference, this incorporates the additional constraint that Luxo be at rest at the beginning and end of motion. Initial values for the orientations were obtained by linear interpolation between the two poses.

The floor enters both as a kinematic constraint and as a force. In general collision constraints appear as inequalities, but to simplify matters, we chose to specify explicitly the time intervals during which Luxo was on the floor, imposing during those times the equality constraints

$$\theta_0 - \frac{\pi}{2} = 0, \mathbf{P} - \mathbf{P}_f = 0$$

where $\theta_0$ is the orientation of the base, $\mathbf{P}$ is the position of the center of the base, and $\mathbf{P}_f$ is a constant point on the floor. In other words, the position and orientation of the base are nailed. The limitation of this formulation, compared to an inequality, is that the times at which contact occurs must be prespecified, rather than allowing things to bounce freely. The floor constraint was enabled for the first and last five frames, allowing time for anticipation and follow-through. Of course, two different values were used for $P_f$ at the start and finish, defining the start end points of the jump.

Unlike the pose constraints, which are really "stage directions," the floor constraint represents a mechanical interaction involving the transmission of force between the base and the floor. This contact force must be taken into account to satisfy the physics constraint. Our collision model has the base colliding with the floor inelastically with infinite friction, which means that the base comes to rest, losing its kinetic energy, at the moment of contact. The contact force is therefore whatever arbitrary force on the base—specifically, on $\mathbf{P}$ and $\theta_0$—is required to satisfy physics in light of the floor constraint. No special provision need be made to solve for the contact forces beyond introducing additional state variables to represent them. Their values are then determined during the constraint-solving process. This method of solving for constraint forces applies to other mechanical constraints, such as joint attachments, and is closely related to the method of Lagrange multipliers.

The choice of optimization criteria is an area we have just begun to explore. Flash et al. [12] propose that unrestrained human arm motions minimize the integral of jerk squared, $\int \left| d^3\mathbf{x}/dt^3 \right|^2 dt$, where $\mathbf{x}$ is the position of the hand. In the examples shown, we chose to optimize a measure of the motion's mechanical efficiency rather than a kinematic smoothness measure. We minimize the *power* consumed by the muscles at each time step, which for each joint is the product of the muscle force and the joint's angular velocity. Our preliminary observation is that this criterion produces more fluid and natural motion than do kinematic smoothness criteria, which often come out looking somewhat arthritic.

Figure 19 shows a series of iterations leading from an initial motion in which Luxo translates, floating well above the floor, to a finished jump in which all the constraints are met and the optimization function is minimized. Note that the elements of realistic motion already appear after the first iteration. The final motion shows marked anticipation, squash-and-stretch, and follow-through. From its predefined initial pose, Luxo assumes a crouch providing a pose from which to build momentum. The crouch is followed by a momentum-building forward-and-upward extension to a stretched launching position. While in flight, the center of mass moves ballistically along a parabolic arc determined by the launch velocity and by the force of gravity. Toward the end of the flight, Luxo once again assumes a crouched position in anticipation of landing, extending slightly while moving toward impact. This "stomp" maneuver has the effect of transferring kinetic energy into the base, where it vanishes in the inelastic collision with the floor. Following impact, luxo extends forward while compressing slightly, dissipating the remaining momentum of flight, then rises smoothly to its prespecified final pose.

In the first variation on the basic jump, we add an additional constraint fixing the contact force on landing. The value we choose provides control over a hard-to-soft landing dimension—a large landing force leads to an exaggerated stomp, as if trying to squash a bug, while a small value leads to a soft landing, as if trying to avoid breaking something fragile. Figure 20 shows a relatively soft landing, generated under the same conditions as the basic jump except for the contact force constraint. Comparing the motion to the basic jump, we see that Luxo softened the blow of impact by squashing while moving toward impact, reducing the velocity, and hence the kinetic energy of the base. In contrast, the basic jump has a small *stretch* before impact, producing an energy-absorbing stomp.

The next variation has the same conditions as the basic jump, but the mass of the base has been doubled. The final motion is shown in Figure 21. As expected, both the anticipation and follow-through are exaggerated in compensation for the greater mass.

A final variation, shown in figure 22, has the conditions of the soft-landing jump, but with a hurdle interposed between start and finish, and an additional constraint that Luxo clear the hurdle. As one would expect, the extra height required is gained by squashing vigorously on approaching the wall.

The jumping examples each took under 10 minutes to compute on a Symbolics 3640. While this is hardly interactive speed, it constitutes a tiny fraction of the cost of high-quality rendering.

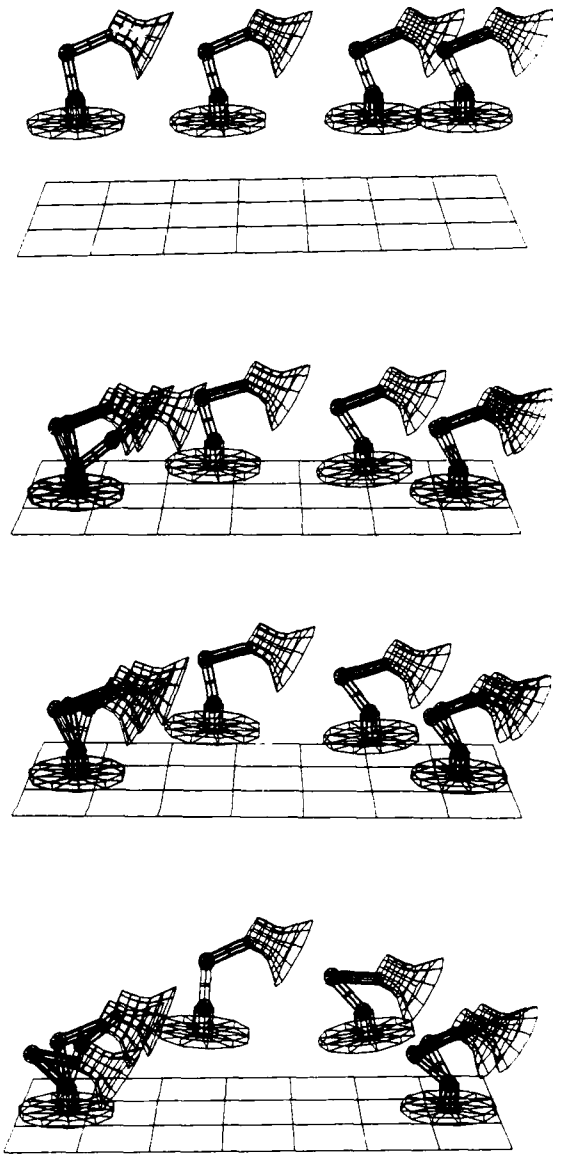


Figure 19: From top to bottom, a series of iterations leading from an initial motion in which Luxo translates, floating above the floor, to a finished jump in which all the constraints are met and the optimization function is minimized. The final motion shows marked anticipation, squash-and-stretch, and follow-through.

**Ski Jumping.** Figure 23 shows Luxo descending a ski jump. As in the previous case, Luxo is constrained to be on the ski jump and the landing at particular time samples. The biggest difference between the ski-jump and the infinite-friction floor of the previous example is that Luxo is free to slide, with the exact positions on the ski jump and the landing left unspecified except at the top and bottom of the ski jump. In addition, there is a constraint that the orientation of the base must be tangent to the surface it is resting on.

Both the ski jump and landing exert forces on Luxo. There is a normal force which keeps him from falling

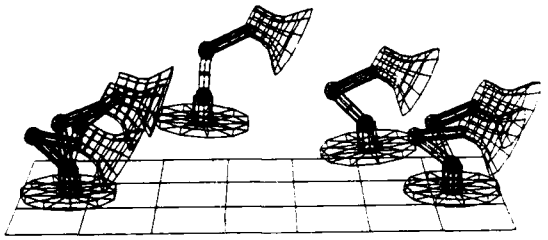


Figure 20: A variation on the basic jump in which the contact force on landing is constrained to be small. The force of impact is reduced by squashing just before landing, reducing the velocity and hence the kinetic energy of the base. In contrast, the jump in figure 19 exhibits a slight *stretch* before impact, producing an energy-absorbing stomp.

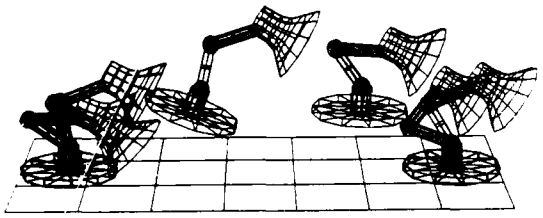


Figure 21: The mass of Luxo's base has been doubled. In other respects, the conditions are the same as those producing the motion in figure 19.

through and a frictional force which is tangent to the surface and proportional to the tangential velocity. The coefficients of friction were state variables in the optimization.

At one time instant while Luxo is in the air, the height of his base is constrained. In addition, there is a term in the objective function which gives him a preference for a particular pose while in the air. This is a "style" optimization without which Luxo is content to go through the air in a bent position.

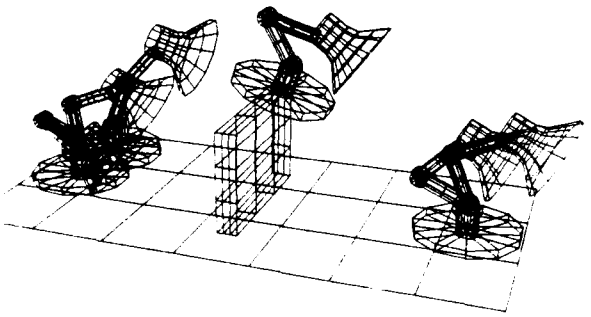


Figure 22: Clearing a hurdle.

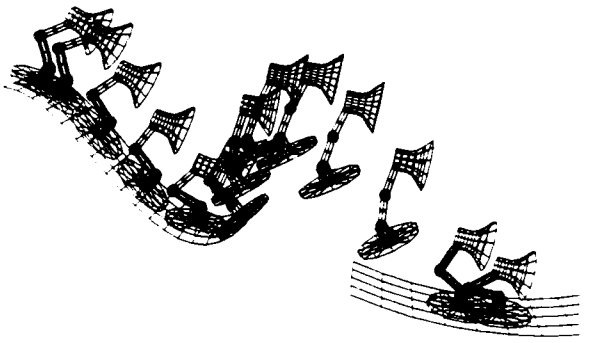


Figure 23: Luxo descends a skijump.

Luxo is also given pose constraints at the beginning and end of the motion. Unlike the previous jumps, however, his initial velocity is unconstrained.

The initial condition for the optimization was a uniform translation in the air above both the ski jump and the landing. In the first iteration, Luxo puts his feet on the ski jump and landing. By iteration 4, there is significant anticipation and follow through. Figure 23 is the result after 16 iterations.

Both the ski jump and landing are built from two B-spline segments. The entire jump was computed with 28 time samples in the optimization. There were 223 constraints and 394 state variables. The Jacobian contained 3587 non-zero entries, about 4% of the total number of entries. The entire motion was computed in 45 minutes on a Symbolics 3600.

## E. Discussion

Our results show that spacetime methods are capable of producing realistic, complex and coordinated motion given only minimal kinematic constraints. Such basic attributes as anticipation, squash-and-stretch, follow-through, and timing emerge on their own from the requirement that the kinematic constraints be met in a physically valid way subject to simple optimization criteria.

The principle advantage of spacetime methods over simple keyframing is that they do much of the work that the animator would otherwise be required to do, and that only a skilled animator *can* do. Motions that would require highly detailed keyframe information may be sketched out at the level of "start here" and "stop there." This is a profoundly different and more economical means of control than conventional keyframing affords, an advantage that easily outweighs the greater mathematical complexity and computational cost of the method.

Beyond sparser keyframing, spacetime methods offer really new forms of motion control. For example, we saw

in the previous section that constraints on forces, such as the force of a collision, can be used in a direct and simple way to say "hit hard" or "hit softly," producing subtle but very effective changes in the motion.

Of the new opportunities for motion control, perhaps the most exciting is the selection of optimization criteria to affect the motion globally, an area we have only begun to explore. With a little thought, it is clear that a magic "right" criterion, whether based on smoothness, efficiency or some other principle, is unlikely to emerge and would in any case be undesirable. This is because the "optimal" way to perform a motion, as with any optimization, depends on what you're trying to do. Consider for example several versions of a character crossing a room: in one case, walking on hot coals; in another, walking on eggs; in another, carrying a full bowl of hot soup; and in still another, pursued by a bear. Plainly the character's goals—and attendant criteria of optimality—are very different in each case. We would hope to see these differing goals reflected in the motion. The possibility of controlling motion directly in terms of its goals, not just where it goes but how, is one we intend to explore.

## References


[1] Armstrong, W. and Green M., *The dynamics of articulated rigid bodies for purposes of animation,* in *Visual Computer,* Springer-Verlag, 1985, pp. 231–240.

[2] Bajcsy, R., and Solina, F., "Three dimensional object representation revisited," *Proc. First Int. Conf. on Computer Vision,* pp. 231–240, London, UK, 1987.

[3] Barzel, R. and Barr, A., *Dynamic constraints,* Topics in Physically Based Modeling, Course Notes, Vol. 16, SIGGRAPH 1987.

[4] Benson, A., and Evans, D.J., *ACM Trans. Mathematical Software* vol. 3, pp 96–103, 1977.

[5] Binford, T. O., "Visual perception by computer," Invited talk IEEE Systems and Control Conf., Miami, FL, 1971

[6] Brady, M., Grimson, W.E.L., Langridge, D., "Shape encoding and subjective contours," Proceedings of American Association for Artificial Intelligence, Stanford University, 1980.

[7] Brady, M., et al., eds, *Robot Motion: Planning and Control,* MIT Press, Cambridge, MA, 1982

[8] Buckley, C., *The Application of Continuum Methods to Path Planning,* Doctoral Dissertation, Dept. of Mechanical Engineering, Stanford University, Stanford, CA, 1985

[9] Burr, D. J. "Elastic matching of line drawings," *IEEE Trans. Pattern Analysis and Machine Intelligence* vol. PAMI-3, p. 708, 1986.

[10] Burt, P. and Julesz, B. "A disparity gradient limit for binocular fusion," *Science* **208** pp. 615-617, 1980.

[11] Fischler, M. A., and Elschlager, R. A. "The representation and matching of pictorial structure," *IEEE Trans. on Computers,* **C-22** p. 67-92, 1973.

[12] Flash,T., and Hogan, N. "The Coordination of Arm Movements: an Experimentally Confirmed Mathematical Model," *MIT AI Memo 786,* November 1984.

[13] Fleischer, K., Witkin, A., Kass, M. and Terzopoulos, D., *Cooking With Kurt,* (video), 1987.

[14] Fleischer, K., *Elastically Deformable Models,* (video), 1987.

[15] Gill, P., Murray, W., and Wright, M., *Practical Optimization,* Academic Press, New York, NY, 1981

[16] Girard, M., and Maciejewski, A., *Computational modeling for the computer animation of legged figures,* Proc. SIGGRAPH, 1985, pp. 263–270

[17] Gladwell, I., and Wait, R., (ed.) *A Survey of Numerical Methods for Partial Differential Equations,* Clarendon, Oxford,1979.

[18] Goldstein, H. *Classical Mechanics,* Addison-Wesley, Reading, MA, 1950

[19] Haumann, D. *Modeling the physical behavior of flexible objects,* Topics in Physically Based Modeling, Course Notes, Vol. 16, SIGGRAPH 1987.

[20] Hildreth, E., "The computation of the velocity field," *Proc. Royal Soc.,* **B221**, pp. 189–220.

[21] Isaacs, P. and Cohen, M., *Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics, Computer Graphics,* **21** (4), July 1987, pp. 215–224 (Proc. SIGGRAPH 1987).

[22] Kanisza, G., "Subjective contours," *Scientific American,* **234**, pp. 48-52., 1976.

[23] Kass, M., and Witkin, A.,"Analyzing oriented patterns," *Computer Vision Graphics and Image Processing* **37**, 362–385, 1987

[24] Kass M., Witkin, A., and Terzopoulos, D., "Snakes: Active contour models." First appeared in *Proc. Intl. Conf. Comp. Vision,* London, 1987. Extended version in *International Journal of Computer Vision* **1** (4) 1987.

[25] Klein, C. and Huang, C., *Review of pseudoinverse control for use with kinematically redundant manipulators,* IEEE Trans. SMC, Vol. 13, No. 3, 1983

[26] Lasseter, J. *Principles of traditional animation applied to 3D computer animation, Computer Graphics,* **21** (4), July 1987, pp. 35–44 (Proc. SIGGRAPH 1987).

[27] Marr, D.,"Analysis of occluding contour,"*Proc. R. Soc. Lond. B* **197** p. 441–475, 1977.

[28] Marr, D. and Nishihara, H. K., "Visual information processing: Artificial intelligence and the sensorium of sight," Technology Review, vol. 81, no. 1, October 1978.

[29] Marr, D. and Poggio, T. "A computational theory of human stereo vision," *Proc. Royal Soc.* **B204**, pp 301-328, 1979.

[30] Marr, D. and Hildreth, E., "A theory of edge detection," *Proc. Royal Soc.*, **B207** pp. 187-217, 1980.

[31] Marr, D., *Vision*, Freeman, San Francisco, 1982.

[32] Martelli, A. "An application of heuristic search methods to edge and contour detection," *CACM* **19** p 73, 1976.

[33] Pentland, A. P., "Parts: Structured descriptions of shape," *Proc. National Conf. on Artificial Intelligence, AAAI-86,* pp. 695–701 Philadelphia, PA, 1986

[34] Pixar, *Luxo, Jr.,* (film,) 1986

[35] Poggio, T. and Torre, V., "Ill-Posed problems and regularization analysis in early vision," Proc. DARPA Image Understanding Workshop, New Orleans, LA, Baumann, Ed., pp. 257-263, 1984.

[36] Poggio, T., Torre, V., and Koch, C., "Computational vision and regularization theory," *Nature,* **317**, 6035 pp 314-319, 1985.

[37] Press, W.. et al., *Numerical Recipes,* Cambridge University Press, Cambridge, England, 1986

[38] Sperling, G. "Binocular vision: A physical and a neural theory," *Am. J. Psychology* **83**, p 461-534, 1970.

[39] Terzopoulos, D., "Regularization of inverse visual problems involving discontinuities," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-8, p. 413–424, 1986.

[40] Terzopoulos, D., and Fleischer, K., "Modeling inelastic deformation: Viscoelasticity, plasticity, fracture," submitted to SIGGRAPH '88.

[41] Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K., "Elastically deformable models," *Computer Graphics,* **21** (4) July 1987 pp. 205–214 (Proc. SIGGRAPH '87).

[42] Terzopoulos, D., Witkin, A., and Kass, M., "Symmetry-seeking models for 3D object reconstruction," First appeared in *Proc. Intl. Conf. Comp. Vision*, London, 1987. Extended version in *International Journal of Computer Vision,* **1** (3), 1987, pp. 211–221.

[43] Terzopoulos, D., Witkin, A., and Kass, M., "Energy constraints on deformable models: recovering shape and nonrigid motion." First appeared in*Proc. AAAI-87*, Seattle. Extended version in press in *Artificial Intelligence.*

[44] Tikhonov, A. N. "Regularization of incorrectly posed problems," *Sov. Math. Dokl.*, vol. 4, pp. 1624-1627, 1963.

[45] Witkin, A. "Scale space filtering," Proceedings of International Joint Conference on Artificial Intelligence, Karlsruhe p. 1019-1021, 1983.

[46] Witkin, A., Terzopoulos, D., and Kass, M. "Signal matching through scale space," *International Journal of Computer Vision,* **1** (2), pp. 133–144, 1987.

[47] Witkin, A., Fleischer, K., and Kass, M. *Knot Reel,* (video), 1986.

[48] Witkin A., Fleischer, K. and Barr, A., "Energy constraints on parameterized models," *Computer Graphics,* **21** (4) July 1987, pp. 225–232 (Proc. SIGGRAPH '87).

[49] Witkin, A., and Fleischer, K., *Energy Constraints on Parameterized Models,* (video), 1987.

[50] Witkin, A. and Kass M., "Spacetime constraints," submitted to SIGGRAPH 1988.

[51] Ullman, S., "Filling the gaps: The shape of subjective contours and a model for their generation," *Biol. Cybernetics* vol. 25, 1976.

[52] Widrow, B. "The rubber mask technique, Parts I and II," *Pattern Recognition*, **5** p. 175-211, 1973.

[53] Wilhelms, J. and Barsky, B., *Using dynamic analysis to animate articulated bodies such as humans and robots*, Graphics Interface, 1985.

[54] Zucker, S., Hummel, R., and Rosenfeld, A., "An application of relaxation labeling to line and curve enhancement," *IEEE Trans. on Computers,* vol. C-26, p. 394, 1977.

[55] Zucker, S. "Computational and psychophysical experiments in grouping: Early orientation selection," in *Human and Machine Vision,* Beck, Hope and Rosenfeld, Eds., pp. 545–567, 1983.

# PERCEPTION WITH FEEDBACK

**Ruzena Bajcsy**
**Computer and Information Science Department**
**University of Pennsylvania**
**Philadelphia, PA 19104**

## ABSTRACT

We have defined active perception as a problem of an intelligent data acquisition process. For that, one needs to define and measure parameters and errors from the scene which in turn can be fed back to control the data acquisition process. This is a difficult though important problem. Why? The difficulty is in the fact that many of the feedback parameters are context and scene dependent. The precise definition of these parameters depends on thorough understanding of the data acquisition devices (camera parameters, illumination and reflectance parameters), algorithms (edge detectors, region growers, 3D recovery procedures) as well as the goal of the visual processing.

**Acknowledgements:** This work is supported in part NSF-CER/DCR82-19196 AO2, NSF-CER MCS-8219196, NSF/DCR-8410771, Air Force/F49620-85-K-0018, ARMY DAAG-29-84-K-0061, DAA29-84-9-0027 and DARPA NOO14-85-K-0807 P002.

## INTRODUCTION

Most past and present work in machine perception has involved extensive static analysis of passively sampled data. However, it should be axiomatic that perception is not passive, but active. Perceptual activity is exploratory, probing, searching; percepts do not simply fall onto sensors as rain falls onto ground. We do not just see, we look. And in the course, our pupils adjust to the level of illumination, our eyes bring the world into sharp focus, our eyes converge or diverge, we move our heads or change our position to get a better view of something, and sometimes we even put on spectacles. This adaptiveness is crucial for survival in an uncertain and generally unfriendly world, as millenia of experiments with different perceptual organizations have clearly demonstrated. Although no adequate account or theory of activity of perception has been presented by machine perception research, very recently, some researchers have recognized the value of actively probing the environment and emphasized the importance of data acquisition during the perception including head/eye movement [Ballard 87, Aloimonos et al 87].

## WHAT IS ACTIVE PERCEPTION?

In the robotics and computer vision literature, the term "active sensor" generally refers to a sensor that transmits (generally electromagnetic or acoustic radiation, e.g., radar, sonar, ultrasound, microwaves and collimated light) into the environment and receives and measures the reflected energy. Active Sensing is included under the term Active Perception. We believe that the use of active sensors is not a necessary condition on active sensing, and that active sensing can be performed with passive sensors (that only receive, and do not emit) employed actively. Here we use the term active not to denote a time-of-flight type sensor, but to denote a passive sensor employed in an active fashion, purposefully changing the sensor's stated parameters according to sensing strategies. Putting it more succinctly, we are introducing a new paradigm for research in Computer Vision [Bajcsy 1985] called Active Perception. The new ingredient of this paradigm is the inclusion of feedback, hence the title of the paper.

### *Problem definition*

The problem of Active Sensing can be stated as a problem to control strategies applied to the data acquisition process that will depend on the current state of the data interpretation including recognition. The question may be asked, "Is Active Sensing only an application of Control Theory?" Our answer is: "No, at least not in its simple version." Here is why:

1. The feedback is performed not only on sensory data but on complex processed sensory data, i.e., various extracted features, including relational features.

2. The feedback is dependent on a priori knowledge --- models that are a mixture of numeric/parametric and symbolic information.

But one can say that Active Perception is an application of intelligent control theory which includes estimation, reasoning, decision making and control. This approach has been eloquently stated by Tenenbaum [Tenenbaum 1970]: "Because of the inherent limitation of a single image, the acquisition of information should be treated as an integral part of the perceptual process...Accommodation attacks the fundamental limitation of image inadequacy rather than the secondary problems caused by it." Although he uses the term *accommodation* rather than *active sensing* the message is the same. Before we can outline the problem of active sensing more formally, we need to enumerate the assumptions under which we are making the design.

### Consequences

The implications of the above realisation are:

1. The necessity of models of sensors and all subsequent processing modules, including noise and uncertainity considerations. This is the decomposition or the analysis part of the system.

2. The models of integration process different modules, including feedback. This is the synthesis portion of the process.

3. Explicit specification of the initial and final state/goal and of the task.

4. Study of the system as a whole, including its stability and performance (what optimization procedures could be and should be applied) and how to constrain the system.

If the Active Perception is a theory, what is its predictive power?

There are three components to our theory. Each with certain predictions:

1. Local models at each processing level, characterised by parameters; these parameters are estimated using estimation theory. These parameters predict the range of values for which this theory works.

2. Evaluation process realized by matching of the results obtained from local models with global models. Here we use optimization theory. The global models will make predictions about the system behavior.

3. Correction and updating, this is the feedback process. As pointed out by Besl and Jain [Besl & Jain 1985] among others, open loop systems are only as robust as their most limited component. So this portion of our theory will predict the amount of improvement due to feedback.

The predictive power of the theory is in the clear definitions of the models/parameters at each level, as well as the global models, i.e., rules of integration or synthesis.

The individual pieces of mathematics used here are not new. What is new is the examination of each model. How appropriate is it to a given task? How valid are the assumptions are and how do the pieces fit together? This paper will try to present those recent pieces of work which address the above issues, the components and the whole of the theory of Active Perception as it applies to 3D shape recovery and description.

## THE MODELS

When we speak about models of sensors we are not restricted to hardware only but also include various software modules that play a role in the processing chain. The following highlights of this work are worth mentioning.

*Sensory Models*

- Physics models -- These models represent the mathematical equations of principles that the sensors operate. The analysis of these models provides range for expected performance of the sensors if no other influences than physics are at work. Examples of these models are optics, illumination, radiance, and forces.

- Geometric models -- From these models we obtain predictions from various aspects of geometry on the best possible values, i.e., the geometry of a pair of stereo cameras predicts how resolution decreases as a function of distance [Solina 1985].

- Signal Models or Ideal Measurement -- Signal models help us analyse and predict the feasibility of detection of certain features. Examples of this case are: edge (step, linear or non-linear) and region (piece-wise constant or linear or nonlinear, but monotonic) models [Haralick & Shapiro 1985, Pavlidis & Liou 1988] as well as shape models of objects.

- Noise or Disturbance Models -- Here we have considered not only the normal distribution (as everybody else has) but also abnormal distributions, symmetric or non-symmetric distributions of the random variables.

All these models provide upper and lower bounds for expected errors, resolution, and robustness, which is necessary for making certain decisions, in particular: "Do we need more data in order to get more accuracy? Can we afford to take more data based on some economy? Given the errors, how do we combine different pieces of information in order to improve the overall performance?"

The Models and Estimation theory have been very successfully applied by Zucker [Zucker 1985]. In this basic work titled: "Theory of Early Orientation Selection", Zucker used the model of a contour that comes from differential geometry. He divides the orientation selection process into three steps:

1. The measurement step-series of convolutions.

2. The interpretation step of these convolution values. This is a functional minimization problem.

3. Finding the integral curve through the vector field.

This decomposition in steps, having the parameters of each step explicit, allows Zucker to make clear predictions about where the contours will or will not be found. We very much agree with Zucker's criticism of the field for lacking this kind of methodology. The very same flavor is in the paper of Leclerc and Zucker [Leclerc & Zucker 1987] where they study the edge detection of image discontinuities. The work of Binford and Nalwa [Nalwa & Binford 1987] is again similar in flavor but is applied to modeling of edges or more general discontinuities.

A systematic and thorough approach to modeling, as it applies to Active Vision, is shown in the recent Ph.D. thesis of E. Krotkov [Krotkov 1987] at the University of Pennsylvania. He has defined the task of determination of spatial layout using an agile camera system and two cues: range from focus and range from vergence. He has decomposed the problem into three subproblems:

1. identifying an appropriate model M to represent the spatial layout of the environment;

2. finding effective methods for constructing M from vision data; and,

3. determining strategies for actively, dynamically, and adaptively setting sensor parameters for acquiring the vision data.

In this section, we shall review only the first subproblem. Krotkov modelled two characteristics of objects: extent and position. This means encoding a map of location of objects with respect to the viewer. The extent was modeled via boxes. In order to accomplish the above he had to model the details of the sensor (the camera) as well as the details of the

computational process of obtaining range from focus and range from vergence. The hardware of the agile camera is in Figure 1. Notice that the physical dimensions of all the components of the hardware are important for determining the physical boundaries of the final result. It is not possible to go into all the details of the analysis [Krotkov 87] but we can summarize the model as follows:

1. determine the optics of the lenses, the depth of field, the accuracy of object distance, (in this setup the distance of the object is independent of the depth of field for distances 1-3 m.)

2. circle of confusion: its diameter depends upon the distance of the object plane from focusing distance. For a given distance between the image and detector planes the confusion circle is directly proportional to the diameter of the aperature, in this case diameter is 58mm.

3. the spatial resolution of the detector array is another limiting factor; (for the CCD chip used in this work the width of one photoreceptor is 0.03 mm and the focal length f=105mm determines the evaluation window size, typically 20x20 pixels).

4. determine how to measure the sharpness of focus with a criterion function. After analysing defocus as an attenuation of high spatial-frequencies and experimentally comparing a number of possible criterion functions, the method based on maximizing the magnitude of the intensity gradient was chosen. It proves superior to others in monotonicity about the mode and in robustness in the presence of noise. Then the Fibonacci search technique is employed to optimally locate the mode of the criterion function.

5. Finally the distance to an object point, given the focus motor position of sharpest focus is modeled by the thick lens law.

All the above predictions were experimentally verified on more than 3,000 points. A very similar exercise that can be presented, although, will not be for lack of space, is the modeling of the physical relationships for the vergence controller and the modeling of the line finder that is being used for matching the two stereo pairs of lines.

## SYSTEMS AND THEIR EVALUATIONS

The word "system" is used in this paper in the very traditional form, that is, anytime two or more modules interact, we speak of a system. We assume, as described above, that each module is clearly defined, characterised by some formal means, i.e., we not only know the input/output function but also the internal parameters (either assumed or estimated). Therefore, the focus of this section is how to model the *interaction* of what we refer to as local models, i.e., *global models*. Therefore, the Global Models must be able to:

1. evaluate the results obtained from local models in the context of the Global expectations or consistency rules; and,

2. make decisions on acceptance/rejection or other actions in the process of perceptual activity.

One can ask: "What is different between this paradigm of Active Perception from the standard robotic paradigm where a mechanical system, i.e., a manipulator or an autonomous land/air/underwater vehicle carries out some actions/tasks based on the sensory input?" The difference is in the task itself. Here, the task is to improve the perceptual process. Such tasks could be to improve the estimation of the location of the robot or of an object; get more or less complete scans of the environment; recognize and classify shape or other properties of the outside world, (discriminate among different substances) etc.

To our knowledge, very little work has been performed on Active Machine Perceptual Systems in the sense of the above definition. To that extent, the state of the art in this domain is poorly defined. In this section, we will briefly describe some of the work most pertinent to our point of view.

A few groups have built hardware/software systems, i.e., sensors that are mobile and under computer control, that enable us to investigate the issues of Active Perception. The hand-eye system at Stanford uses a pan-tilt head, a lens turred for controlling focal length, color and neutral filters mounted on a "filter wheel", and a vidicon camera whose sensitivity is programmable. This system is limited in processing, i/o speed (it used a PDP-6 computer) and in resolution (four bits), but its design is well conceived. It was successfully used for adaptive edge following [Tenenbaum 1971].

POPEYE is a grey level vision system developed at Carnegie Mellon University [Bracho et al]. It is a loosely coupled multiprocessor system on a MULTIBUS, with a MC68000, a Matrix frame grabber and buffer, as an array processor, dedicated image processing units, and a programmable transform processor. Image positioning is achieved with a pan/tilt head and motorized zoom/focus lens. Altogether, this is a powerful and flexible system. Weiss [Weiss 1984] describes a model reference adaptive control feedback system that uses image features (areas, centroids) as feedback control signals. We would like to do something similar, using a world model rather than image features.

Kuno et al [Kuno et al 1985] reports a stereo camera system whose interocular distance yaw, and tilt are computer controlled. The cameras are mounted on a specially designed linkage. By controlling the interocular distance a certain flexibility in processing is achieved: the larger the distance, the more precisely disparity information from stereo can be converted to absolute distance information; the smaller the distance, the easier is the solution to the correspondence problem. It is not clear how this flexibility will be exploited, nor how the three degrees of freedom will be controlled.

Recently, the Rochester group [Ballard 87] as well as Poggio at MIT [Poggio 87] have reported a mobile stereo camera. Both these groups are interested in modeling the biological systems rather than the machine perception problems.

*Models for Segmentation*

Local Models for Segmentation

"What do we mean by Local models in this case?" Usually one begins with local (small neighborhood) edge/region estimation. The problem here is the choice of scale. "At which scale should one be detecting discontinuities (edges) and continuities (regions)?" Typically, the scale for detecting edges and regions is different. This is also the approach taken by us [Anderson et al. 1987]. The block diagram of this system is shown in Figure.. In the feedback we control the similarity criterion of the region growing procedure. The stopping rule is satisfied by a close value of GOODNESS MEASURE. (GOODNESS = Number of edge-border confirmed pixels - 0.5 unconfirmed border pixels and constant 0.5 is bias toward oversegmentation made by us based on experience) Figure 3 is an example of a rejected partition segmentation as oversegmentation. Figures 4 a) is the original image and 4b) is the best segmentation complying to the goodness measure. An alternative approach has been developed by Mallat [1987] in that having all the power of two scales available simultaneously.

The effect of this is for every edge point we have a vector representation of zero-crossings at several scales. Then the segmentation is formulated as a problem of local aggregation based on the distance measure between vectors. We have tested this approach so far only on few examples, which seem to gave promise.

Global Models for Segmentation

The most popular approach in this domain is the cooperative network or relaxation approach where the global model is the continuity or some other consistency principle. This is represented by various workers, notably Rosenfeld, Zucker, & Faugeras, [Rosenfeld et al 1987, Faugeras & Berthod 1981]. Another recently popular global model is the Random Markow Fields model, used by several researchers [Geman & Geman 1984, Derin & Won 1987, Cross & Jain 1983]. These models have been used for texture and image segmentation in general. The principle of this model is that the effects of members of the field upon each other are limited to local interaction as defined by the neighborhood. This is also the weakness of the model. This basically assumes that we a priori know the spatial arrangements, their surface properties and illumination of objects. This assumption is too strong and applicable only in a few, highly controlled experiments. The important point here is at which level one chooses models. We advocate spatially (especially positionally) invariant models, i.e., topological models together with several and different surface/region models. [Anderson et al 1987]

A very interesting global model has been inspired by the influence of the Gestalt psychological models implemented in Reynolds and Beveridge [Reynolds & Beveridge 1987]. The global model is a geometric grouping system based on similarity and spatial proximity of geometrical tokens. Examples for lines are: colinearity, rectangularity, and so on. We think that this is a promising approach, but what is missing is evaluation criterion and robustness studies. This cannot be avoided especially with such vague notions as similarity and spatial (or other) proximity relations.

The group at the University of Massachusetts [Kohl et al 1987] have been advocating, for some time, that one must be goal directed in order to do low level image processing and/or segmentation. They argue that the failure of general segmentation techniques can be traced to:

1. the image is too complex because of the physical situation from which the image was derived and/or the nature of the scene; or,
2. there is a problem of evaluating different region/line segmentations.

They say: "...it has been our experience that no low-level evaluation measure restricted to making measurements on the segmentation can provide a useful comparative metric. Rather the quality of segmentation can be measured only with respect to the goals of an interpretation..."

We agree that the images are complex, but some of the image acquisition process can be modeled, and hence, one can account for the variability of the acquisition process as shown by Krotkov [Krotkov 87]. Of course, one cannot predict the spatial arrangement of objects, (their surface properties) but we can have models that are somewhat invariant to these variables. We also agree with the authors that there are no good segmentation evaluation functions. It is known that segmentation process is not unique given any number of parameters. But we wish to argue with the authors that the only thing that can determine the segmentation is the goal of its interpretation. If this would be so, then we cannot ever have a general (or even semi-general) purpose system which can bootstrap itself and adapt to an a priori unknown environment. We hypothesis that there are several levels of global models going from more general, context-independent to more specific, context- and domain-dependent. In reality, they do present an intermediate system called GOLDIE which indeed has several syntactic (context-independent) evaluation functions. This is quite encouraging although it still must be tested in a variety of domains. (It was tested only in one domain, sofar). Perhaps this is a good place to point out the need by the researchers to turn to a priori contextual knowledge in order to cope with the complexity problem of segmentation (as an example), and hence, constrain the nondeterminism of the process which can be resolved by an opposite approach that the Active Perception paradigm offers, that is, to get more data which can resolve the multiplicity of interpretations.

This point has recently been presented in a paper at the First International Conference on Computer Vision by Aloimonos and Badyopadhyay [Aloimonos & Badyopadhyay 1987] with the title: "Active Vision." They argue that: "problems that are *ill-posed, nonlinear* or unstable for a passive observer become well-posed, linear or stable for an active observer." They investigated five typical computer vision problems: shape from shading, shape from contour, shape from texture, structure from motion and optic flow (area based). The principal assumption that they make is that the active observer moves with a known motion, and of course, has available more than one view, i.e., more data. Hence, it is not surprising that with more measurements taken in a controlled fashion, ill-posed problems get converted into well-posed problems.

*Model for 3D Shape*

Local Models

As with the problem of segmentation, the local models for 3D shape are primarily surface and/or silhouette models. Given the three dimensional data points (obtained by various means: laser range finder, stereo, range from focus, range from motion, etc.) There are several ways in the literature (see the overview, Besl & Jain 1987) how one can estimate local surface properties and classify them into planar, curved with different curvature characteristics. The problem with local methods is that they are frequently too noise sensitive.

Global Models

One global model can be and is in this example an analytic function, superquadric defined below:

$$F(x, y, z) = \left[ \left[ \left( \frac{x}{a_1} \right)^{\frac{2}{\epsilon_2}} + \left( \frac{y}{a_2} \right)^{\frac{2}{\epsilon_2}} \right]^{\frac{\epsilon_2}{\epsilon_1}} + \left( \frac{z}{a_3} \right)^{\frac{2}{\epsilon_1}} \right]^{\epsilon_1}$$

Expanding this formula for general position and orientation, one can use fitting procedures to estimate from the data the parameters that in this case represent: position, orientation of the object, its main dimensions, $(a_1, a_2, a_3)$ and so called shape parameters $(E_1, E_2)$, i.e., the roundness or squareness of the edges. Adding some global deformations to the model, parameters along some principle axis, such as bending, tapering and twisting parameters can be recovered too. This problem of fitting in a seventeen-parameter space has been investigated by F.Solina in his Ph.D. dissertation [Solina 1987]. The appeal of this work is not only that it provides parameters that have a natural mapping to shape descriptors and that it covers a large class of objects, but also that the evaluation function, which in this case is the least squares estimation, gives us a numerical value of how well the data has been explained via this model.

A similar methodology is the use of variational calculus for estimating a given geometric surface or volumetric model. Examples of this work are Grimson and Horn.

The conclusive message of this section is clear: The evaluation criterion is necessary in order to have a predictive theory. In turn, we cannot have an evaluation criterion without a global model.

INTEGRATION OF STRATEGIES

It is not so difficult to see that any reasonable 3D shape recognition system will have to have at least three representations: volume, surface and silhouette or boundary. The volume representation will give us the global information while the surface and contour will give us more the details of the shape. Hence what one wants is a system which will have all these processes work simultaneously and integrate the results! The heart of this problem is: to identify the purpose/goal of the process, have explicit model (as described above) of all the volume extraction, the surface and silueht or boundary extraction, and evaluate which of the processes

(volume,surface,boundary) is fitting (the best, fastest or other optimization criterion) the goal. We are building such a system in the GRASP laboratory at the University of Pennsylvania. Since all our models are being evaluated via fitting procedures, we have an explicit measure of how well each process explains the data. This measure in turn will be used as the confidence measure for the integration process.

A very simple and preliminary example of this kind of interaction is shown in Figure 5. Figure 6 shows the segmented range image of an envelope and estimated its surface and contour, simultaneously fitted superquadric model.

## CONCLUSIONS

In conclusion we have defined active perception as a problem of an intelligent data acquisition process. For that, one needs to define and measure parameters and errors from the scene which in turn can be fed back to control the data acquisition process. This is a difficult though important problem. Why? The difficulty is in the fact that many of the feedback parameters are context and scene dependent. The precise definition of these parameters depends on thorough understanding of the data acquisition devices (camera parameters, illumination and reflectance parameters), algorithms (edge detectors, region growers, 3D recovery procedures) as well as the goal of the visual processing. The importance however of this understanding is that one does not spend time on processing and artificially improving imperfect data but rather on accepting imperfect, noisy data as a matter of fact and incorporating it into the overall processing strategy.

Why has it not been pursued earlier? The usual answers one gets are: Lack of understanding of static images, the need to solve simpler problems first, less data, etc. This of course is a misconception. One view lacks information which is laborously recovered when more measurements, i.e., more views can resolve the problem easier. More views, adds a new dimension, time which requires new understanding, new techniques and new paradigms.

## REFERENCES


Aloimonos, J..Weiss, I. and Badyopadhyay, A., "Active Vision," *Proceedings Darpa Image Understanding Workshop*, pp. 552-573, Los Angeles, (February 1987).

Aloimonos, J. and Badyopadhyay, A. "Active Vision," *First Int. Conf. on Computer Vision*, IEEE, pp. 35-54, (June 1987).

Anderson, H., "Edge-Detection for Object Recognition in Aerial Photographs," University of Pennsylvania, Grasp Laboratory, Technical Report MS-CIS-87-96, 1987.

Anderson, H., Bajcsy R. and Mintz, M., "A Modular Feedback System for Image Segmentation," *University of Pennsylvania, Grasp Laboratory*, TR-110, (1987).

Bajcsy, R., "Active Perception vs. Passive Perception," *In Third IEEE Workshop on Vision*, pp. 55-59, Bellaire, (1985).

Ballard, D. H., "Eye Movements and Spatial Cognition," *Computer Science Department*, University of Rochester, TR-218, (November 1987).

Besl, P. and Jain, R., "Three-Dimensional Object Recognition", *Computing Surveys*, Vol.17, No.1, March 1985.

Bracho, R., Schlag, J. F. and Sanderson, A. C., "POPEYE: A Gray-Level Vision System for Robotics Applications," *CMU-RI-TR-83-6*, (May, 1983).

Cross, G. R. and Jain, A. K., "Markow Random Field Texture Models," *IEEE Trans.* **PAMI**, Vol. 5, pp. 25-39, (1983).

Derin, H. and Won, Ch. S. "A Parallel Image Segmentation Algorithm Using Relaxation with Varying N Neighborhoods and its Mapping to Array Processors," *CVGIP*, Vol. 40, No. 1, pp. 54-78, (October 1987).

O. Faugeras and M. Berthold, "Improving consistency and reducing ambiguity in stochastic labeling: An optimization approach", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-3, p. 245, 1981.

Geman, S. and Geman, D., "Stochastic Relaxation, Gibbs Distributing and the Bayesian Restoration of Images," *IEEE Trans.* **PAMI** Vol. 6, pp. 721-741, (November 1984).

Haralick, R.M. and Shapiro, L. G., "Image Segmentation Techniques," *Computer, Vision, Graphics, and Image Processing*, 29(1), pp.100-133 (January 1985).

Kohl, Ch. A, Hanson, A. R. and Riseman, E. M. "Goal Directed Control of Low-Level Processes for Image Interpretation," *IU Proceedings* **DARPA**, Vol. 2, pp. 538-551, (February 1987).

Krotkov, E. "Exploratory Visual Sensing for Determining Spatial Layout with an Agile Stereo Camera System," *University of Pennsylvania Ph.D. Dissertation also available as a Technical Report MS-CIS-87-29*, (April 1987).

Kuno, Y., Numagami H. , Ishikawa, M., Hoshino, H. and Kidode, M., "Three-Dimensional Vision Techniques for an Advanced Robot System," *IEEE Conference on Robotics and Automation*, pp. 11-16, St. Louis, (March, 1985).

Leclerc, Y. G. and Zucker, S. W. "The Local Structure of Image Discontinuities in One Dimension," *IEEE Trans.* **PAMI**, Vol. 9, Num. 3, pp. 341-355, (May 1987).

Mallat, S.G., A Theory for Multiresolution Signal Decomposition: the Wavelet Representation", Technical Report, MS-CIS-87-22, University of Pennsylvania, May 1987.

Nalwa, V. S. and Binford, T. O., "On Detecting Edges," *IEEE Trans.* **PAMI-8**, No. 6, pp. 699-714, (November 1986).

Pavlidis, R and Liou, Y.T., "Integrating Region Growing and Edge Detection", submitted to *ICVPR*, (1988).

Poggio, T., "MIT Progress in Understanding Images," *Proceedings of DARPA IU Workshop*, pp. 41-54, Los Angeles, (February 1987).

Reynolds, G. and Beveridge J. R. "Searching for Geometric Structure in Image of Natural Scenes," *Image Understanding Proceedings* **DARPA**, Vol. I, pp. 257-271, (February 1987).

Rosenfeld, A., Hummel, R. A. and S. U. Zucker, "Scene Labelling by Relaxation Operations," *IEEE Trans. SMC G*, pp. 420-433, (1976).

Solina, F., "Errors in Stereo Due to Quantization," *University of Pennsylvania TR-MS-CIS-85-34*, (December, 1985).

Solina, F., "Shape Recovery and Segmentation with Deformable Part Model," *University of Pennsylvania Ph.D. Dissertation also available as a Technical Report TR-MS-CIS-87-111*, (December 1987).

Tenenbaum, J. M., "Accommodation in Computer Vision," *Stanford University Ph.D. Thesis*, (November, 1970).

Tenenbaum, J. M., "A Laboratory for Hand-Eye Research," *IFIPS*, pp. 206-210, (1971).

Weiss, L. E., "Dynamic Visual Servo Control of Robots: An Adaptive Image-Based Approach," *CMU-RI-TR-84-16*, (1984).

Zucker, S. W., "Early Orientation Selection: Tangent Fields and the Dimensionality of their Support," *Computer Vision & Robotics Laboratory*, Department of Electrical Engineering, *McGill University*, Montreal, Quebec, Canada, *TR-85-13-R*, (May 16, 1985).

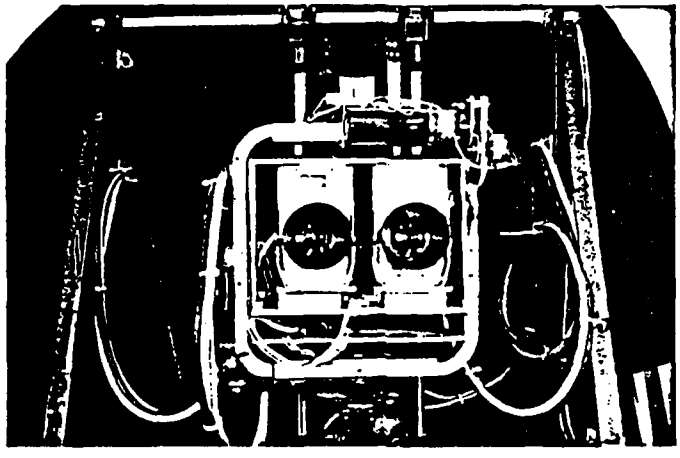


Figure 1: Photograph of the Camera System

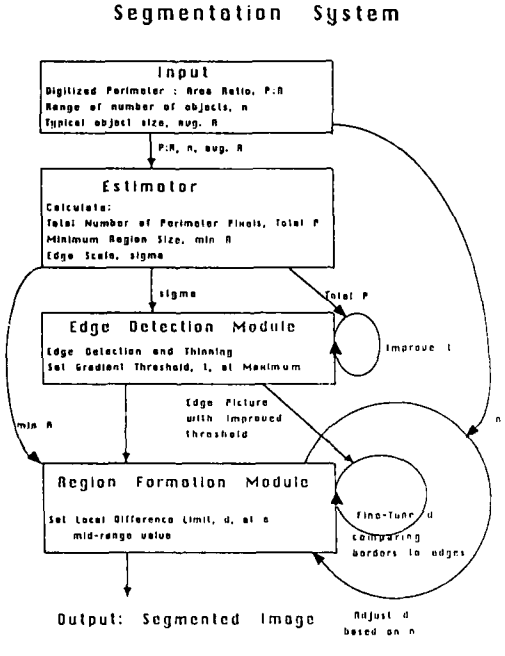
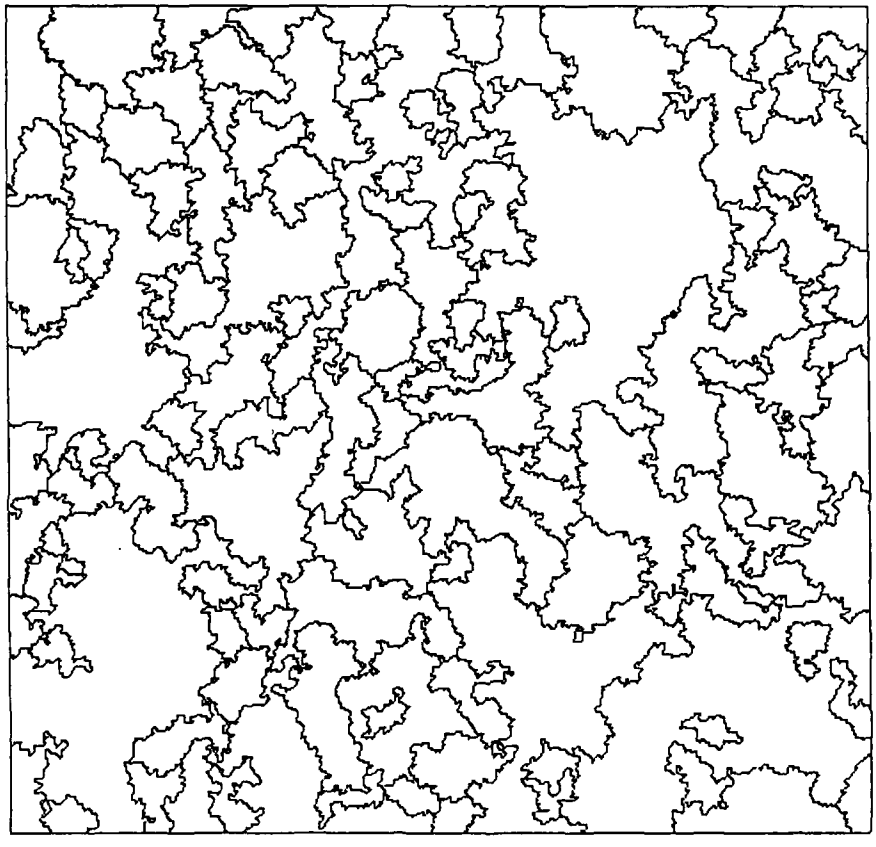



Figure 2



Segmented Scene of Outdoor Aerial Photo

Figure 3

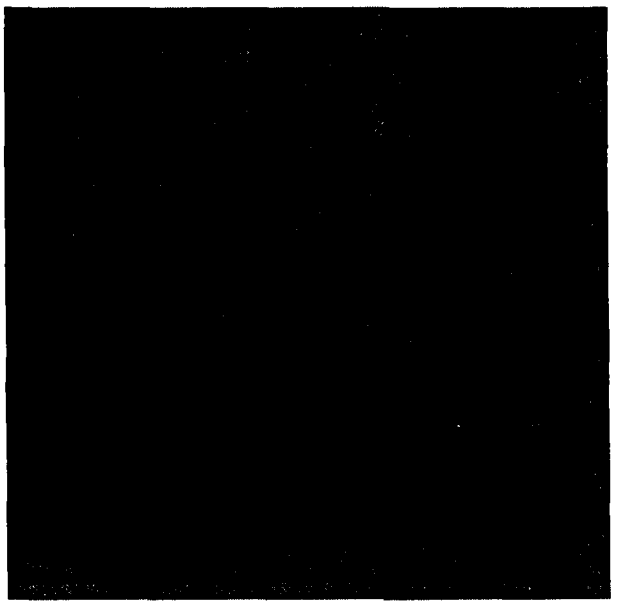


Figure 4a
Image data used for segmentation produced from original image f pennies

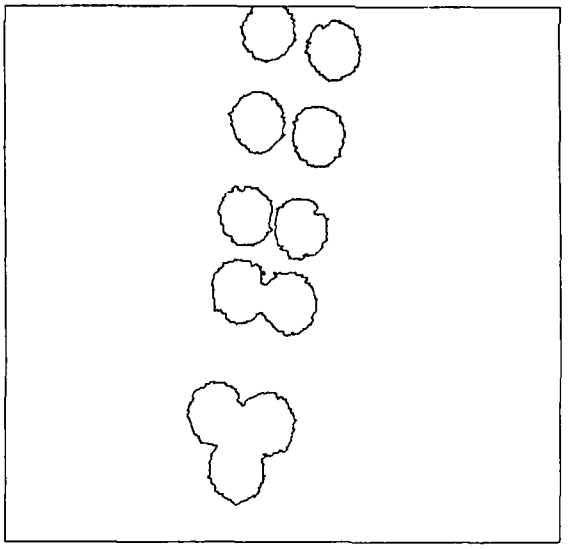


Figure 4b

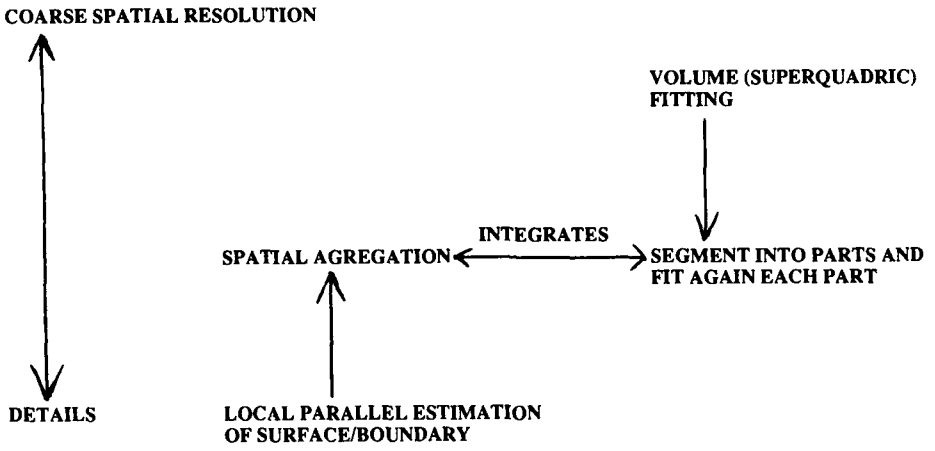
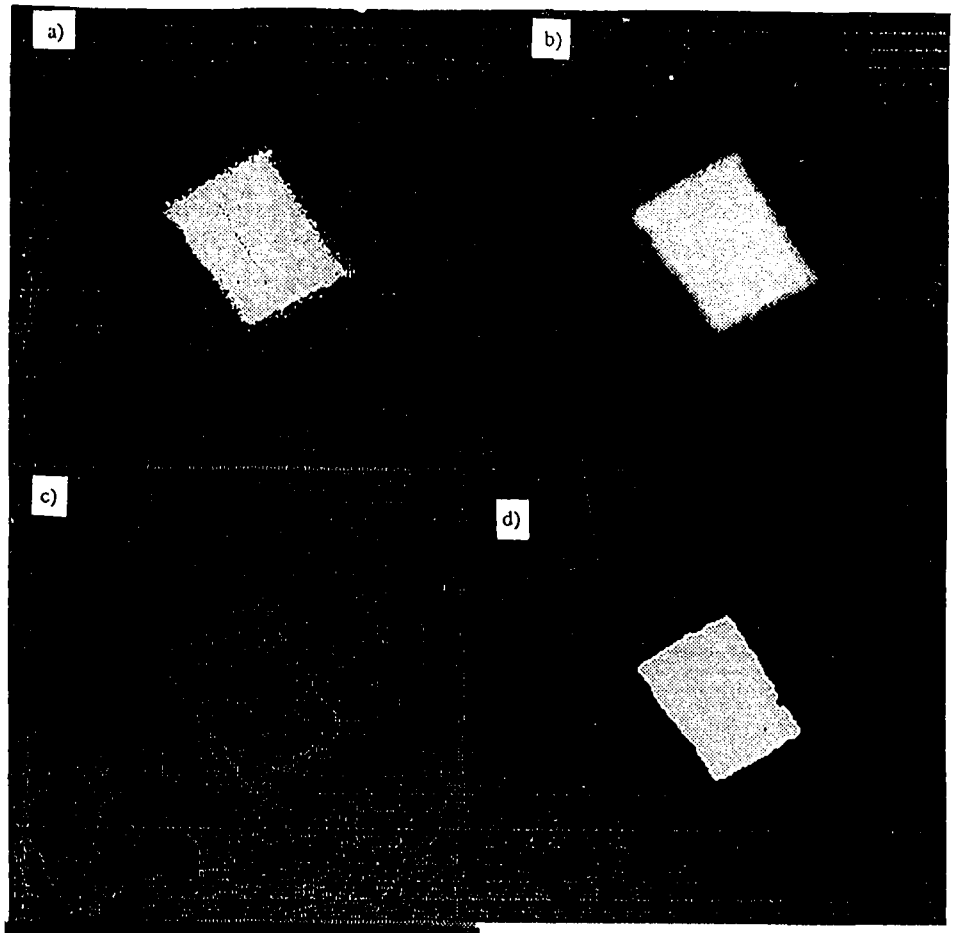



Figure 5: Proposed Integrated System for 3-D Shapes



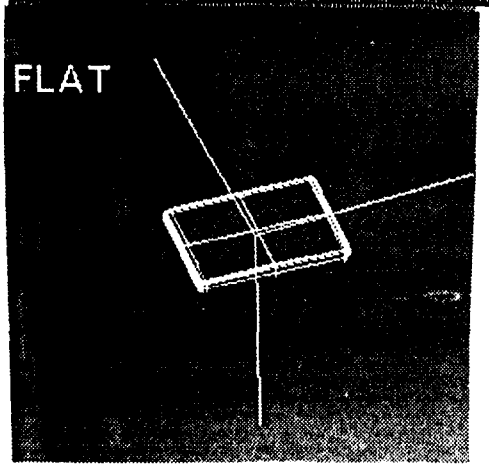




Figure 6: a) Input range image
b) Smoothed range image
c) Depth discontinuity (thresholded)
d) To-most object detected
e) Flat

# Qualitative Motion Detection and Tracking of Targets from a Mobile Platform

**Bir Bhanu and Wilhelm Burger**

Honeywell Systems and Research Center
3660 Technology Drive, Minneapolis, MN 55418

## ABSTRACT

The problem of understanding scene dynamics is to find consistent and plausible 3-D interpretations for any change observed in the 2-D image sequence. Due to the motion of the Autonomous Land Vehicle (ALV), stationary objects in the scene generally do not appear stationary in the image, whereas moving objects are not necessarily seen in motion. The three main tasks of our novel approach for target motion detection and tracking are: (a) to estimate the vehicle's motion, (b) to derive the 3-D structure of the stationary environment, and (c) to detect and classify the motion of individual targets in the scene. These three tasks strongly depend on each other. The direction of heading (i.e. translation) and rotation of the vehicle are estimated with respect to stationary locations in the scene. The focus of expansion (FOE) is not determined as a particular image location, but as a region of possible FOE-locations called the *Fuzzy FOE*. We present a qualitative strategy of reasoning and modeling for the perception of 3D space from motion information. Instead of refining a single quantitative description of the observed environment over time, multiple *qualitative interpretations* are maintained simultaneously. This offers superior robustness and flexibility over traditional numerical techniques which are often ill-conditioned and noise-sensitive. A rule-based implementation of this approach is discussed and results on real ALV imagery are presented.

## 1. INTRODUCTION

Visual information is an indispensable clue for the successful operation of an Autonomous Land Vehicle (ALV). Even with the use of sophisticated inertial navigation systems, the accumulation of position errors requires periodic corrections. Operation in unknown environments or mission tasks involving search, rescue, or manipulation critically depend upon visual feedback.

Assessment of scene dynamics becomes vital when moving objects may be encountered, e.g., when the ALV follows a convoy, approaches other vehicles, or has to detect moving threats. For the given case of a moving camera, image motion can also supply important information about the spatial layout of the environment ("motion stereo") and the actual movements of the ALV. This is a valuable input for navigation and vehicle control, i.e., steering, accelerating, and braking.

Previous work in motion analysis has mainly concentrated on numerical approaches for the recovery of 3D motion and scene structure from 2D image sequences. Recently, Nagel[11] gave an excellent review. The most common approach is to estimate 3D structure and motion in one computational step by solving a system of linear or non-linear equations.[5,17] This technique is characterized by several severe limitations.

First, it is known for its notorious noise-sensitivity. To overcome this problem, some researchers have extended this technique to cover multiple frames.[3,7] Secondly, it is designed to analyze the relative motion and 3D structure of a single rigid object. To estimate the ALV's egomotion and the scene structure, the environment would have to be treated as a large rigid object. However, rigidness of the environment cannot be guaranteed due to the possible presence of moving objects in the scene. But what is the consequence of accidentally including a moving 3D point into the system of equations? In the best case, the solution (in terms of motion and structure) would exhibit a large residual error, indicating some non-rigid behavior. The point in motion, however, cannot be immediately identified from this solution alone. In the worst case (for some forms of motion), the system may converge towards a rigid solution (with small error) in spite of the actual movement in the point set. This again shows another (third) limitation: there is no suitable means of expressing the ambiguity and uncertainty inherent to dynamic scene analysis.

The approach that we propose is novel in two important aspects. First, scene structure is not treated as a mere by-product of the motion computation but as a valuable means to overcome some of the ambiguities of dynamic scene analysis. The key idea is to use the description of the scene's 3D structure as a link between motion analysis and other processes that deal with spatial perception, such as shape-from-occlusion, stereo, spatial reasoning, etc. A 3D interpretation of a moving scene can only be correct if it is acceptable by all the processes involved.

Secondly, numerical techniques have been largely replaced by a qualitative strategy of reasoning and modeling. The use of qualitative techniques in computer vision has been of growing interest recently.[16,19] Basically, instead of having a system of equations approach a single rigid (but possibly incorrect) numerical solution, we maintain multiple qualitative interpretations of the scene. All the existing interpretations are kept consistent with the observations made in the past. The main advantage of this approach is that a new interpretation can be supplied immediately when the currently favored interpretation turns out to be unplausible.

These interpretations are built in three separate steps (see Figure 1). First, significant features (points, boundaries, corners, etc.) are extracted from the image and the 2D dis-

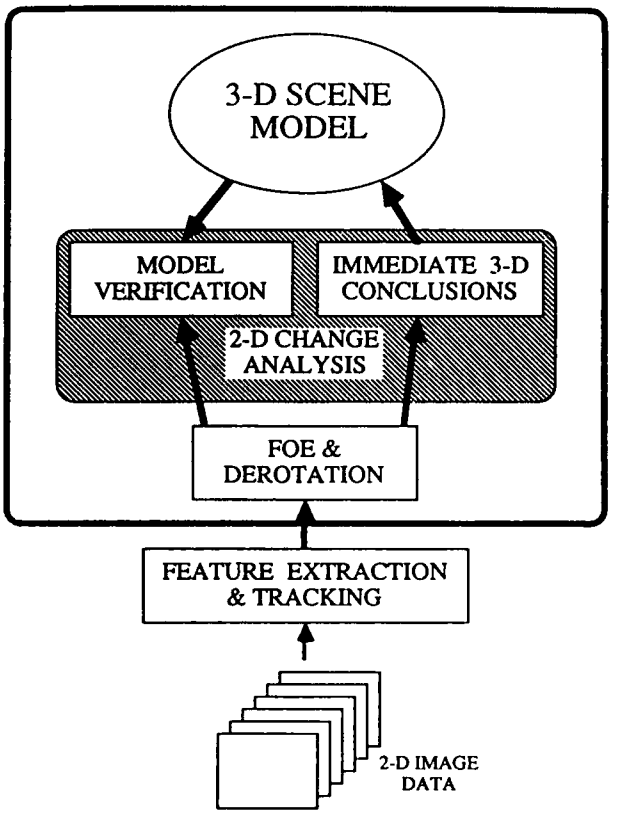



Figure 1. Main steps of the Qualitative Motion Detection and Tracking approach.

placement vectors are computed for this set of features. For the examples shown here, points were selected and tracked between individual frames. Automatic techniques suitable for this task can be found elsewhere.[2, 10] In the second step, the vehicle's direction of translation, i.e. the Focus of Expansion (FOE), and the amount of rotation in space are determined. The effects of vehicle motion on the FOE computation is described in section 2. Almost all the necessary numerical computation is performed in the FOE computation stage, which is described in section 3. The third step (2D Change Analysis) constructs an internal 3D model of the scene. Section 4 outlines the concepts and operation of this Qualitative Scene Model. Experiments with our approach on real imagery taken from the moving ALV are discussed in section 5. Finally, section 6 presents the conclusions of the qualitative motion detection and tracking system.

## 2. EFFECTS OF VEHICLE MOTION

The first step of our approach is to estimate the vehicle's motion relative to the stationary environment using visual information. Arbitrary movement of an object in 3D space and thus the movement of the vehicle itself can be described as a combination of translation and rotation. While knowledge about the composite vehicle motion is essential for control purposes, only translation can supply information about the spatial layout of the 3D scene (motion stereo). This, however, requires the removal of all image effects resulting from vehicle rotation. For this purpose, we discuss the changes upon the image that are caused by individual application of the "pure" motion components.

### 2.1 Viewing Geometry

It is well-known that any rigid motion of an object in space between two points in time can be decomposed into a combination of translation and rotation. While many researchers[13, 15] have used a velocity-based formulation of the problem, the following treatment views motion in discrete time steps.

Given the world coordinate system (X Y Z) shown in Figure 2, a translation $\mathbf{T} = (U\ V\ W)^T$ applied to a point in 3D $\mathbf{X} = (X\ Y\ Z)^T$ is accomplished through vector addition:

$$\mathbf{X}' = \mathbf{T} + \mathbf{X} = \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} U \\ V \\ W \end{bmatrix} + \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{1}$$

A 3D rotation **R** about an arbitrary axis through the origin of the coordinate system can be described by successive rotations about its three axes:

$$\mathbf{R} = \mathbf{R}_\phi\ \mathbf{R}_\theta\ \mathbf{R}_\psi \tag{2}$$

where

$$\mathbf{R}_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \text{ rotation about the X–axis,} \tag{3a}$$

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \text{ rotation about the Y–axis,} \tag{3b}$$

$$\mathbf{R}_\psi = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ rotation about the Z–axis.} \tag{3c}$$

A general rigid motion in space consisting of translation and rotation is described by the transformation

$$\mathbf{M}: \mathbf{X} \rightarrow \mathbf{X}' = \mathbf{R}_\phi\ \mathbf{R}_\theta\ \mathbf{R}_\psi\ (\mathbf{T}+\mathbf{X}) \tag{4}$$

Its six degrees of freedom are $U$, $V$, $W$, $\phi$, $\theta$ and $\psi$ .

This decomposition is not unique because the translation could be as well applied after the rotation. Also, since the multiplication of the rotation matrices is not commutative, a different order of rotations would result in different amounts of rotation for each axis. For a fixed order of application, however, this motion decomposition is unique.

To model the movements of the vehicle, the camera is considered as being stationary and the environment as being moving as one single rigid object relative to the camera. The origin of the coordinate system is located in the lens center of the camera.

### 2.2 Image Effects of 3D Camera Motion

The given task is to reconstruct the vehicle's egomotion from visual information. It is therefore necessary to know the effects of different kinds of vehicle motion upon the camera image.

Under perspective imaging, a point in space $\mathbf{X} = (X\ Y\ Z)^T$ is projected onto a location on the image plane $\mathbf{x} = (x\ y)^T$ such that

$$x = f\frac{X}{Z} \qquad y = f\frac{Y}{Z}\ , \tag{5}$$

where $f$ is the focal length of the camera (see Figure 2).

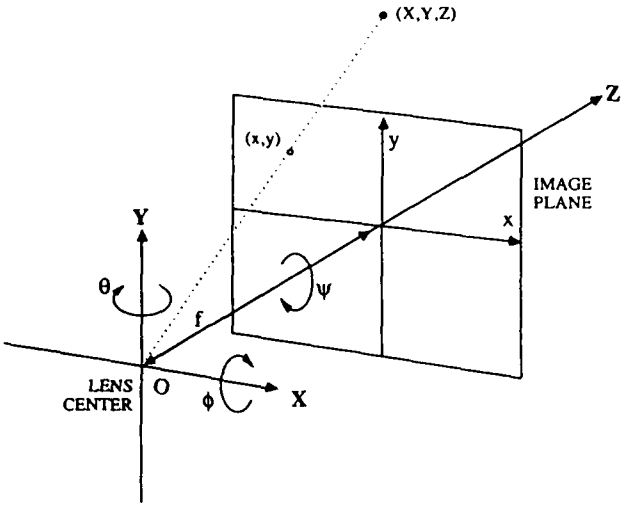



Figure 2: Camera Model showing the coordinate system, lens center, image plane and angles of rotation. The origin of the coordinate system is located at the lens center. The focal length $f$ is the distance between the lens center and the image plane.

### 2.2.1 *Effects of Pure Camera Rotation*

When the camera is rotated around its lens center, the acquired image changes but no new views of the environment are obtained. Camera rotation merely maps the image into itself.

The most intuitive effect results from pure rotation about the Z-axis of the camera-centered coordinate system, which is also the optical axis. Any point in the image moves along a circle centered at the image location $\mathbf{x} = (0\ 0)$.

In practice, however, the amount of rotation of the vehicle about the Z-axis is small. Therefore, vehicle rotation is confined to the X- and Y-axis, where significant amounts of rotation occur.

The vehicle undergoing rotation about the X-axis by an angle $-\phi$ and the Y-axis by an angle $-\theta$ moves each 3D point $\mathbf{X}$ to point $\mathbf{X}'$ relative to the camera.

$$\mathbf{X} \rightarrow \mathbf{X}' = \mathbf{R}_\phi \cdot \mathbf{R}_\theta \cdot \mathbf{X} \tag{6}$$

$$= \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ \sin\phi \sin\theta & \cos\phi & -\sin\phi \cos\theta \\ -\cos\phi \sin\theta & \sin\phi & \cos\phi \cos\theta \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}.$$

Consequently $\mathbf{x}$, the image point of $\mathbf{X}$, moves to $\mathbf{x}'$ given by

$$x' = f \frac{X \cos\theta + Z \sin\theta}{-X \cos\phi \sin\theta + Y \sin\phi + Z \cos\phi \cos\theta} \tag{7a}$$

$$y' = f \frac{X \sin\phi \sin\theta + Y \cos\phi - Z \sin\phi \cos\theta}{-X \cos\phi \sin\theta + Y \sin\phi + Z \cos\phi \cos\theta} \tag{7b}$$

Inverting the perspective transformation for the original image point $\mathbf{x}$ yields

$$X = \frac{1}{f} Z x \quad \text{and} \quad Y = \frac{1}{f} Z y . \tag{8}$$

The 2D *rotation mapping* $\mathbf{r}_\phi \mathbf{r}_\theta$ which moves each image point $\mathbf{x} = (x\ y)$ into the corresponding image point $\mathbf{x}' = (x'y')$ under camera rotation $\mathbf{R}_\phi \mathbf{R}_\theta$ (i.e., a particular sequence of *pan* and *tilt*) is given by

$$\mathbf{R}_\phi \mathbf{R}_\theta\ (\mathbf{X}) : \mathbf{X} \rightarrow \mathbf{X}'$$

$$\mathbf{r}_\phi \mathbf{r}_\theta\ (\mathbf{x}) : \mathbf{x} = (x\ y) \rightarrow \mathbf{x}' = (x'\ y')$$

$$x' = f \frac{x \cos\theta + f \sin\theta}{-x \cos\phi \sin\theta + y \sin\phi + f \cos\phi \cos\theta} \tag{9a}$$

$$y' = f \frac{x \sin\phi \sin\theta + y \cos\phi - f \sin\phi \cos\theta}{-x \cos\phi \sin\theta + y \sin\phi + f \cos\phi \cos\theta} . \tag{9b}$$

It is important to notice that this transformation contains no 3D variables and is therefore a mapping of the image onto itself. This demonstrates that no additional information about the 3D structure of the scene can be obtained under pure camera rotation.

An interesting property of this mapping should be mentioned at this point, which might not be obvious. Moving an image point on a diagonal passing through the center of the image at 45° by only rotating the camera does *not* result in equal amounts of rotation about the X- and the Y-axis. This is again a consequence of the successive application of the two rotations $\mathbf{R}_\theta$ and $\mathbf{R}_\phi$, since the first rotation about the Y-axis also changes the orientation of the camera's X-axis in 3D space. It also explains why the pair of equations in (7) is not symmetric with respect to $\theta$ and $\phi$.

### 2.2.2 *Measuring the Amount of Camera Rotation*

The problem to be solved is the following: Given are two image locations $\mathbf{x}_0$ and $\mathbf{x}_1$, which are the observations of the same 3D point at time $t_0$ and time $t_1$. What is the amount of rotation $\mathbf{R}_\phi$ and $\mathbf{R}_\theta$ which applied to the camera between time $t_0$ and time $t_1$, would move image point $\mathbf{x}_0$ onto $\mathbf{x}_1$ assuming that no camera translation occurred at the same time?

If $\mathbf{R}_\phi$ and $\mathbf{R}_\theta$ are applied to the camera separately, the points in the image move along hyperbolic paths.[12] If pure *horizontal* rotation were applied to the camera, a given image point $\mathbf{x}_0$ would move on a path described by

$$\mathbf{r}_\theta\ (\mathbf{x}_0): y^2 = y_0^2 \frac{f^2 + x^2}{f^2 + x_0^2} . \tag{10}$$

Similarly pure *vertical* camera rotation would move an image point $\mathbf{x}_1$ along

$$\mathbf{r}_\phi\ (\mathbf{x}_1): x^2 = x_1^2 \frac{f^2 + y^2}{f^2 + y_1^2} . \tag{11}$$

Since the 3D rotation of the camera is modeled as being performed in two separate steps ( $\mathbf{R}_\theta$ followed by $\mathbf{R}_\phi$ ), the *rotation mapping* $\mathbf{r}_\phi \mathbf{r}_\theta$ can also be separated into $\mathbf{r}_\theta$ followed by $\mathbf{r}_\phi$. In the first step, applying pure (horizontal) rotation around the Y-axis $\mathbf{r}_\theta$, point $\mathbf{x}_0$ is moved to an intermediate image location $\mathbf{x}_c$. The second step, applying pure (vertical) rotation around the X-axis $\mathbf{r}_\phi$, takes point $\mathbf{x}_c$ to the final image location $\mathbf{x}_1$ . This can be expressed as

$$\mathbf{r}_{\phi\theta} = \mathbf{r}_\phi\ \mathbf{r}_\theta \quad \text{,where} \tag{12}$$

$$\mathbf{r}_\theta: \mathbf{x}_0 = (x_0\ y_0) \rightarrow \mathbf{x}_c = (x_c\ y_c)$$

$$\mathbf{r}_\phi: \mathbf{x}_c = (x_c\ y_c) \rightarrow \mathbf{x}_1 = (x_1\ y_1) .$$

As shown in Figure 3, the image point $\mathbf{x}_c = (x_c\ y_c)$ is the intersection point of the hyperbola passing through $\mathbf{x}_0$ result-

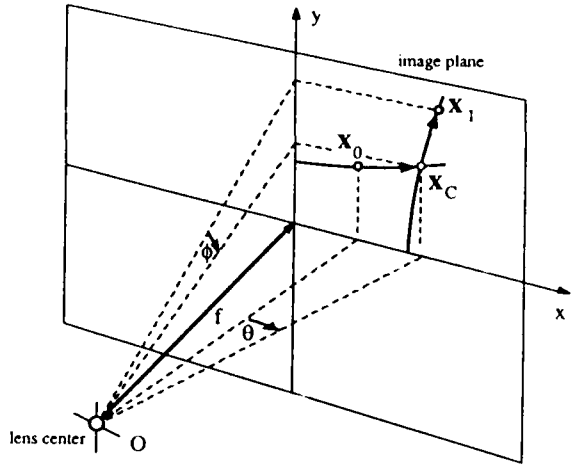



Figure 3: Successive Application of Horizontal and Vertical Rotation. The image point $\mathbf{x}_0$ is to be moved to location $\mathbf{x}_1$ by pure horizontal and vertical camera rotation. Horizontal rotation (about the Y-axis) is applied first, moving $\mathbf{x}_0$ to $\mathbf{x}_C$, which is the intersection point of the two hyperbolic paths for horizontal and vertical rotation. In a second step $\mathbf{x}_c$ is taken to $\mathbf{x}_1$. Then the two rotation angles $\theta$ and $\phi$ are found directly.

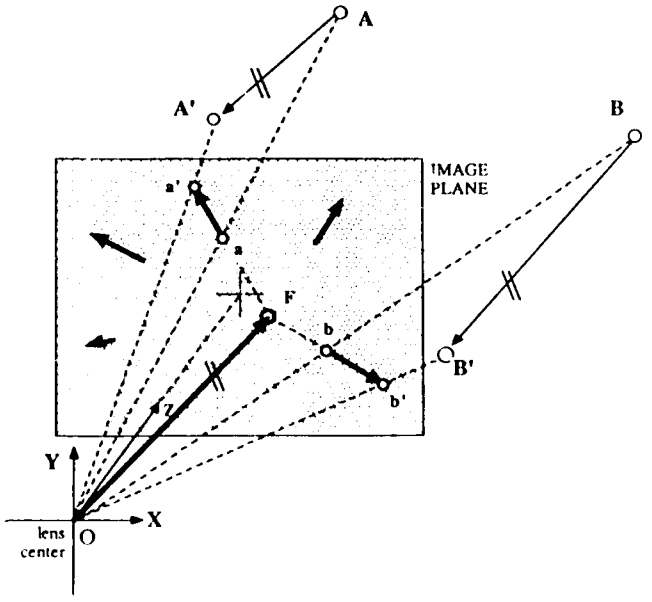



Figure 4: Location of the focus of expansion (FOE). With pure vehicle translation, points in the environment (A, B) move along 3D vectors parallel to the vector pointing from the lens center to the FOE in the camera plane. These vectors form parallel lines in space which have a common vanishing point (the FOE) in the perspective image.

ing from horizontal camera rotation (10) with the hyperbola passing through $\mathbf{x}_1$ resulting from vertical camera rotation (11). Intersecting the two hyperbolae gives the image point $\mathbf{x}_c$, with

$$x_c = f\, x_1 \left[ \frac{f^2 + x_0^2 + y_0^2}{(f^2 + x_0^2)(f^2 + y_1^2) - x_1^2\, y_0^2} \right]^{1/2} \tag{13a}$$

$$y_c = f\, y_0 \left[ \frac{f^2 + x_1^2 + y_1^2}{(f^2 + x_0^2)(f^2 + y_1^2) - x_1^2\, y_0^2} \right]^{1/2} . \tag{13b}$$

The amount of camera rotation necessary to map $\mathbf{x}_0$ onto $\mathbf{x}_1$ by applying $\mathbf{R}_\theta$ followed by $\mathbf{R}_\phi$ is finally obtained as

$$\theta = \tan^{-1} \frac{x_c}{f} - \tan^{-1} \frac{x_0}{f} \tag{14}$$

$$\phi = \tan^{-1} \frac{y_c}{f} - \tan^{-1} \frac{y_1}{f} . \tag{15}$$

### *2.2.3 Effects of Pure Camera Translation*

When the vehicle undergoes pure translation between time $t$ and time $t'$, every point on the vehicle is moved by the same 3D vector $\mathbf{T} = (U\ V\ W)^T$. Again, the same effect is achieved by keeping the camera fixed and moving every point $\mathbf{X}_i$ in the environment to $\mathbf{X}_i'$ by applying $-\mathbf{T}$ .

Since every stationary point in the environment undergoes the same translation relative to the camera, the imaginary lines between corresponding points $\overline{\mathbf{X}_i\mathbf{X}_i'}$ are parallel in 3D space.

It is a fundamental result from perspective geometry[4] that the images of parallel lines pass through a single point in the image plane called a *vanishing point*. When the camera moves along a straight line, every (stationary) image point seems to expand from this vanishing point or contract towards it when the camera moves backwards. This particular image location is therefore commonly referred to as the *Focus of Expansion* (FOE) or the *Focus of Contraction* (FOC). Each displacement vectors passes through the FOE creating the typical radial expansion pattern shown in Figure 4.

As can be seen in Figure 4, the straight line passing through the lens center of the camera and the FOE is also parallel to the 3D displacement vectors. Therefore, the 3D vector $\overrightarrow{\mathbf{OF}}$ points in the direction of camera translation in space. Knowing the internal geometry of the camera (i.e., the focal length), the direction of vehicle translation can be determined by locating the FOE in the image. The actual translation vector $\mathbf{T}$ applied to the camera is a multiple of the vector $\overrightarrow{\mathbf{OF}}$ which supplies only the *direction* of camera translation but not its *magnitude*. Therefore,

$$\mathbf{T} = \lambda\, \overrightarrow{\mathbf{OF}} = \lambda\, |\, x_f\ y_f\ f\, |^T \quad , \lambda \in R\ . \tag{16}$$

Since most previous work incorporated a velocity-based model of 3D motion, the *Focus of Expansion* has commonly been interpreted as the *direction of instantaneous heading*, i.e., the direction of vehicle translation during an infinitely short period in time. When images are given as "snapshots" taken at discrete instances of time, the movements of the vehicle must be modeled accordingly as discrete movements from one position in space to the next.

Therefore, the FOE cannot be interpreted as the momen-

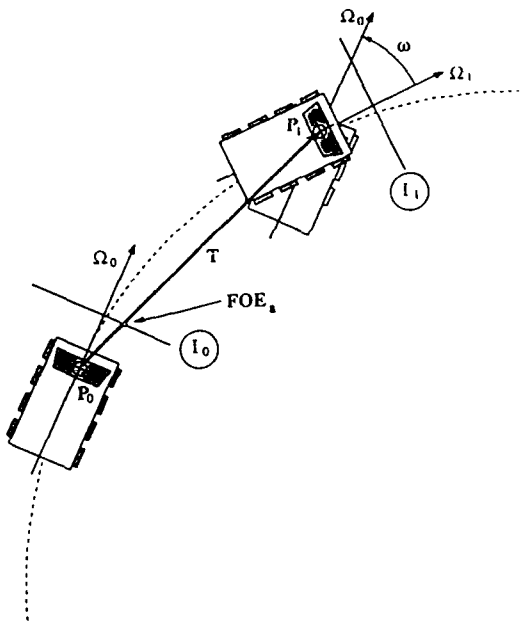



Figure 5: Concept of the FOE for discrete time steps. The vehicle's motion between two points in time can be decomposed into a translation followed by a rotation: the image effects of pure translation ($FOE_a$) are observed in image $I_0$. This scheme is used throughout this work.

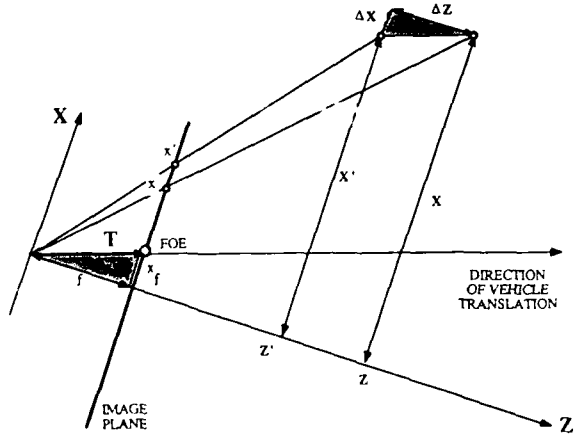



Figure 6: Amount of expansion from the FOE for discrete time steps. The camera moves by a vector **T** in 3D space, which passes through the lens center and the FOE in the camera plane. The 3D **Z**-axis is also the optical axis of the camera.

tary direction of translation at a certain point in time, but rather as the direction of *accumulated* vehicle translation over a *period* of time.

Figure 5 shows the top view of a vehicle traveling along a curved path at two instances in time $t_0$ and $t_1$. The position of the vehicle in space is given by the position of a reference point on the vehicle **P** and the orientation of the vehicle $\Omega$. Figure 5 also displays the adopted scheme of 3D motion decomposition: First the translation **T** is applied which shifts the vehicle's reference point (i.e., the lens center of the camera) from position $\mathbf{P}_0$ to position $\mathbf{P}_1$ without changing the vehicle's orientation $\Omega_0$. The 3D translation vector intersects the image plane at $FOE_a$. In the second step the vehicle is rotated by $\omega$ to the new orientation $\Omega_1$. Translation **T** transforms image $I_0$ into image $\mathbf{I}_1'$, which again is transformed into $\mathbf{I}_1$ by rotation $\omega$. The important fact is that $FOE_a$ is observed at the transition from image $\mathbf{I}_0$ to image $\mathbf{I}_1'$, which is obtained by *derotating* image $\mathbf{I}_1$ by $-\omega$. Throughout the rest of this work, this scheme (Figure 5) is used as a model for vehicle motion.

### *2.2.4 Measuring the Amount of Camera Translation*

Figure 6 shows the geometric relationships for the 2D case. It can be considered as a top view of the camera, i.e., a projection onto the X/Z-plane of the camera-centered coordinate system. The cross section of the image plane is shown as a straight line. The camera is translating from left to right in the direction given by $\mathbf{T} = (x_f \; f)^T$.

A stationary 3D point is observed at two instances of time, which moves in space relative to the camera from **X** to **X'**, resulting in two images $x$ and $x'$.

$$\mathbf{X} = \begin{bmatrix} X \\ Z \end{bmatrix} \quad \text{and} \quad \mathbf{X}' = \begin{bmatrix} X' \\ Z \end{bmatrix} = \begin{bmatrix} X - \Delta X \\ Z - \Delta Z \end{bmatrix}. \tag{17}$$

Using the inverse perspective (8) transformation yields

$$Z = \frac{f}{x} X \quad \text{and} \tag{18}$$

$$Z' = Z - \Delta Z = \frac{f}{x'} X' = \frac{f}{x} (X - \Delta X).$$

From similar triangles (shaded in Figure 6)

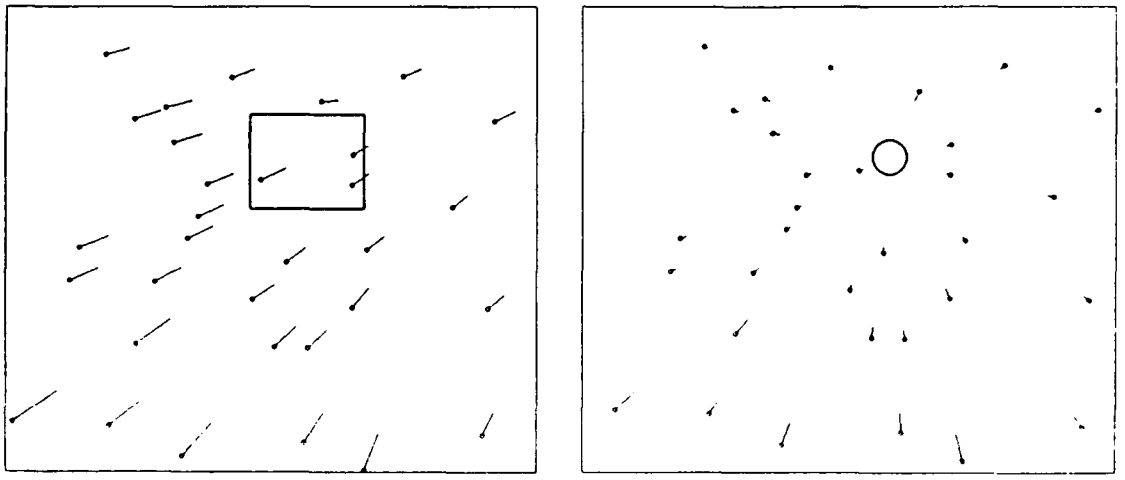
$$\frac{\Delta X}{x_f} = \frac{\Delta Z}{f}, \tag{19}$$

and therefore

$$Z = \Delta Z \frac{x' - x_f}{x' - x} = \Delta Z \left[ 1 + \frac{x - x_f}{x' - x} \right]. \tag{20}$$

Thus, the rate of expansion of image points from the FOE contains direct information about the distance of the corresponding 3D points from the camera. Consequently, if the vehicle is moving along a straight line and the FOE has been located, the 3D structure of the scene can be determined from the expansion pattern in the image. However, the distance *Z of a 3D point from the camera can only be obtained* up to the scale factor $\Delta Z$, which is the distance that the vehicle advanced along the Z-axis during the elapsed time.

When the velocity of the vehicle ($\Delta Z / t$) in space is known, the absolute range of any stationary point can be computed. Alternatively, the velocity of the vehicle can be obtained if the actual range of a point in the scene is known (e.g., from laser range data). In practice, of course, any such technique requires that



*(a)* *(b)*

Figure 7: Simulated displacement field caused by a combination of horizontal and vertical rotation and vehicle translation. *(a)* The rectangle marks the area of search for the FOE. *(b)* The *derotated* displacement with the FOE marked by a circle.

- the FOE can be located in a small area, and
- the observed image points exhibit significant expansion away from the FOE.

As will be shown in the following section, imaging noise and camera distortion pose serious problems in the attempt to assure that both of the above criteria are met.

If a set of stationary 3D points $\{(\mathbf{X}_i\ \mathbf{X}_i')\}$ is observed, then of course the translation in the Z-direction is the same for every point.

$$Z_i - Z_i' = Z_j - Z_j' = \Delta Z \quad \text{for all } i,j\ . \tag{21}$$

Therefore, the range of every point is proportional to the observed amount of expansion of its image away from the FOE

$$Z_i = \propto \frac{x_i' - x_f}{x_i' - x_i}\ , \tag{22}$$

which renders the relative 3D structure of the set of points.

The effects of camera translation **T** can be formulated as a mapping **t** of a set of image locations $\{\mathbf{x}_i\}$ into another set of image locations $\{\mathbf{x}_i'\}$. Unlike in the case of pure camera rotation, this mapping not only depends upon the 3D translation vector but also upon the actual 3D location of each individual point observed. Therefore, in general, **t** is *not* simply a mapping of the image onto itself.

However, one important property of **t** can be described exclusively in the image plane, namely that each point must map onto a straight line passing through the original point and one unique location in the image (the FOE). This means that if the vehicle is undergoing pure translation, then there must exist an image location $\mathbf{x}_f$ such that the mapping **t** satisfies the condition

*radial-mapping* $\mathbf{t}\ (\mathbf{x}_f, \mathbf{I}, \mathbf{I}')$ :

$$\mathbf{t} = \{\ (\mathbf{x}_i\ ,\ \mathbf{x}_i') \in \mathbf{I}\times\mathbf{I}' \mid \mathbf{x}_i' = \mathbf{x}_i + \mu_i\ (\mathbf{x}_i - \mathbf{x}_f)\ , \tag{23}$$

$$\mu_i \in R,\ \mu_i \geq 0\ \}\ .$$

*2.2.5 Combined Effects of Translation and Rotation*

When the vehicle is not undergoing pure translation or rotation but combined 3D motion of the form $\mathbf{R}_\phi\ \mathbf{R}_\theta\ \mathbf{T}$, the effects in the image are described by a transformation **d** (for *displacement*) which is a combination of $\mathbf{r}_\phi$, $\mathbf{r}_\theta$ and **t**:

$$\mathbf{d} : \mathbf{I} \rightarrow \mathbf{I}' = \mathbf{r}_\phi\ \mathbf{r}_\theta\ \mathbf{t}\ (\mathbf{I})\ , \tag{24}$$

where $\mathbf{I} = \{\ \mathbf{x}_i\ \}$, $\mathbf{I}' = \{\ \mathbf{x}_i'\ \}$ are the two sets of corresponding image points.

Figure 7 shows a typical displacement field for a camera undergoing horizontal and vertical rotation as well as translation. The points $\mathbf{x}_i \in \mathbf{I}$ are marked with small circles.

By decomposing a composite displacement field **d** into its three components $\mathbf{r}_\phi$, $\mathbf{r}_\theta$, and **t**, the vehicle's rotation and direction of translation in space can be computed from the information available in the image. This problem is addressed in the following section.

## 3. DECOMPOSITION OF IMAGE MOTION

### 3.1 Problem Statement

As discussed in the previous section, the 3D motion **M** of the vehicle is modeled by a translation **T** followed by a rotation $\mathbf{R}_\theta$ about the Y-axis and a rotation $\mathbf{R}_\phi$ about the X-axis:

$$\mathbf{M} = \mathbf{R}_\phi\ \mathbf{R}_\theta\ \mathbf{T}\ . \tag{25}$$

This results in a mapping $\mathbf{d}$ from the original image $\mathbf{I}_0$ at time $t_0$ into the new image $\mathbf{I}_1$ at time $t_1$.

$$\mathbf{d} : \mathbf{I}_0 \rightarrow \mathbf{I}_1 = \mathbf{r}_\phi \, \mathbf{r}_\theta \, \mathbf{t} \, \mathbf{I}_0 = \mathbf{r}_\phi \, \mathbf{r}_\theta \, \mathbf{I}_0' . \quad (26)$$

The intermediate image $\mathbf{I}_0'$ in (26) is the result of the translation component of the vehicle's motion and has the property of being a radial mapping (23). Unlike the two images $\mathbf{I}_0$ and $\mathbf{I}_1$, which are actually given, the image $\mathbf{I}_0'$ is generally not observed, except when the camera rotation is zero. It serves as an intermediate result to be reached during the separation of translational and rotational motion components.

The question at this point is whether there exists more than one combination of rotation mappings $\mathbf{r}_\phi$ and $\mathbf{r}_\theta$ which would satisfy this requirement, i.e., if the solution is *unique*. It has been pointed out in the previous section that the decomposition of 3D motion into $\mathbf{R}_\phi$, $\mathbf{R}_\theta$, $\mathbf{R}_\psi$, and $\mathbf{T}$ is unique for a fixed order of application. This does not imply, however, that the effects of 3D motion upon the perspective image are unique as well.

Tsai and Huang[17] have shown that seven points in two perspective views suffice to obtain a unique interpretation in terms of rigid body motion and structure, except for a few cases where points are arranged in some very special configuration in space. Ullman[18] reports computer experiments which suggest that six points are sufficient in many cases and seven or eight points yield unique interpretations in most cases.

Due to its design and the application, however, the motion of the ALV in space is quite restricted. The vehicle can only travel upright on a surface and its large wheelbase allows for only relatively small changes in orientation. It is also heavy and thus exhibits considerable inertia. Therefore, the final motion parameters must lie within a certain narrow range and it can be expected that a unique solution can be found even in cases when the number of points is near the above minimum.

The fact that

$$\mathbf{I}_0' = \mathbf{r}_\theta^{-1} \, \mathbf{r}_\phi^{-1} \, \mathbf{I}_1 = \mathbf{t} \, \mathbf{I}_0 \quad (27)$$

suggests different strategies for separating the motion components:

(1) *FOE from Rotation*: Successively apply combinations of inverse rotation mappings $\mathbf{r}_{\theta_1}^{-1} \, \mathbf{r}_{\phi_1}^{-1}$, $\mathbf{r}_{\theta_2}^{-1} \, \mathbf{r}_{\phi_2}^{-1}$,... $\mathbf{r}_{\theta_k}^{-1} \, \mathbf{r}_{\phi_k}^{-1}$ to the second image $\mathbf{I}_1$, until the resulting image $\mathbf{I}'$ is a radial mapping with respect to the original image $\mathbf{I}_0$. Then locate the FOE $\mathbf{x}_{f_k}$ in $\mathbf{I}_0$.

(2) *Rotation from FOE*: Successively select FOE-locations (different directions of vehicle translation) $\mathbf{X}_{f_1}$, $\mathbf{X}_{f_2}$,... $\mathbf{X}_{f_l}$ in the original image $\mathbf{I}_0$ and then determine the inverse rotation mapping $\mathbf{r}_{\theta_l}^{-1} \, \mathbf{r}_{\phi_l}^{-1}$ that yields a radial mapping with respect to the given FOE $\mathbf{x}_{f_l}$ in the original image $\mathbf{I}_0$.

Both alternatives were investigated under the assumption of restricted, but realistic vehicle motion, as stated earlier. It turned out that the major problem in the *FOE-from-Rotation* approach is to determine if a mapping of image points is (or is close to being) radial when the location of the FOE is unknown. Of course, in the presence of noise, this problem becomes even more difficult. The second approach was examined after it appeared that any method which extends the given set of displacement vectors *backwards* to find the FOE is inherently sensitive to image degradations.

Although there have been a number of suggestions for FOE-algorithms in the past,[8,12,15] no results of implementations have been demonstrated on real outdoor imagery. One reason for the absence of useful results might be that most researchers have tried to locate the FOE in terms of a single, distinct image location. In practice, however, the noise generated by merely digitizing a perfect translation displacement field may keep the resulting vectors from passing through a single pixel. Even for human observers it seems to be difficult to determine the exact direction of heading (i.e., the location of the FOE on the retina). Average deviation of human judgement from the real direction has been reported[14] to be as large as 10° and up to 20° in the presence of large rotations.

It was, therefore, an important premise in this work that the final algorithm should determine an *area* of potential FOE-locations (called the *Fuzzy FOE*) instead of a single (but probably incorrect) point.

### 3.2 FOE from Rotation

In this method, the image motion is decomposed in two steps. First, the rotational components are estimated and their inverses are applied to the image, thus partially "derotating" the image. If the rotation estimate was accurate, the resulting displacement field after derotation would diverge from a single image location (the FOE). The second step verifies that the displacement field is actually radial and determines the location of the FOE. For this purpose, two problems have to be solved:

(1) how to estimate the rotational motion components without knowing the exact location of the FOE,

(2) how to measure the "goodness of derotation" and locate the FOE.

#### *3.2.1 Estimating the rotational components*

Each vector in the displacement field is the sum of vector components caused by camera rotation and camera translation. Since the displacement caused by translation depends on the depth of the corresponding points in 3D space (equation 18), points located at a large distance from the camera are not significantly affected by camera translation. Therefore, one way of estimating vehicle rotation is to compute $\theta$ and $\phi$ from displacement vectors which are *known* to belong to points at far distance. Under the assumption that those displacement vectors are only caused by rotation, equations 14 and 15 can be applied to find the two angles. In some situations, distant points are selected easily. For example, points on the horizon are often located at a sufficient distance from the vehicle. Image points close to the axis of translation would be preferred because they expand from the FOE slower than other points at the same depth.

However, points at far distances may not always be available or may not be *known* to exist in the image. In those cases, the following method for estimating the rotational components can be used. The design of the ALV (and most other mobile robots) does not allow rapid changes in the direction of vehicle heading. Therefore, it can be assumed that the motion of the camera between two frames is constrained, such that the FOE can change its location only within a certain range. If the FOE was located in one frame, the FOE in the subsequent frame must lie in a certain image region around the previous FOE location.

Figure 8(a) illustrates this situation. The FOE of the previous frame was located at the center of the square, thus the FOE in the given frame must be inside this square.

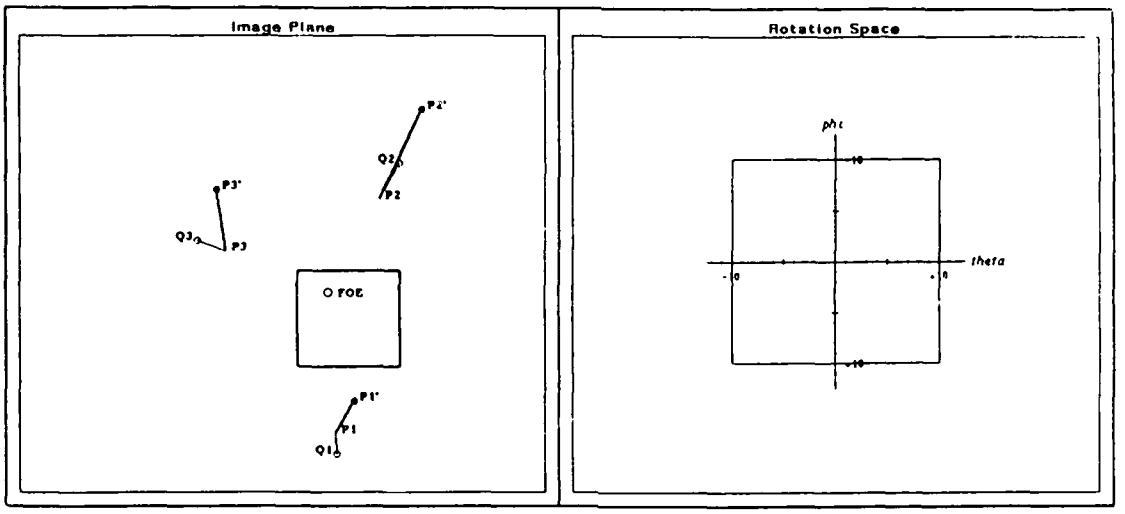



Figure 8: Image Plane and Rotation Space. The displacement field in the Image Plane *(a)* contains three vectors ($P1 \rightarrow P1'$, $P2 \rightarrow P2'$, $P3 \rightarrow P3'$). The previous FOE was observed at the center of the square, which outlines the region of search for the current FOE. The translational displacement components ($P1 \rightarrow Q1$, $P2 \rightarrow Q2$, $P3 \rightarrow Q3$) and the current location of the FOE are unknown but marked in this picture. The initial range of possible camera rotations is ±10° in either direction, indicated by a square in Rotation Space *(b)*.

Three displacement vectors are shown ($P1 \rightarrow P1'$, $P2 \rightarrow P2'$, $P3 \rightarrow P3'$). The translational components ($P1 \rightarrow Q1$, $P2 \rightarrow Q2$, $P3 \rightarrow Q3$) of those displacement vectors and the FOE (inside the square) are not known at this point in time.

The main idea of this technique is to determine the possible range of camera rotations which would be consistent with the FOE lying inside the marked region. Since the camera rotates about two axes, the resulting range of rotations can be described as a *region* in a 2D space. Figure 8(b) shows this *Rotation Space* with the two axes *theta* and *phi* corresponding to the amount of camera rotation around the Y-axis and the X-axis respectively. The initial rotation estimate is a range of ±10° in both directions which is indicated by a square in rotation space.

In general, the range of possible rotations is described by a closed, convex polygon in rotation space. A particular rotation $(\theta', \phi')$ is possible if its application to every displacement vector (i.e., to its endpoint) yields a new vector which lies on a straight line passing through the maximal FOE-region. The region of possible rotations is successively constrained by applying the following steps for every displacement vector (Figure 9):

*(a)* Apply the rotation mapping defined by the *vertices* of the rotation polygon to the endpoint $\mathbf{P}'$ of the displacement vector. This yields a set of image points $\tilde{\mathbf{P}}_i$.

*(b)* Connect the points $\tilde{\mathbf{P}}_i$ to a closed polygon in the image. This polygon is similar to the rotation polygon but distorted by the nonlinear rotation mapping (Figure 9(a)).

*(c)* Intersect the polygon in the image with the open triangle formed by the starting point $\mathbf{P}$ of the displacement vector and the two tangents onto the FOE-region. The result is a new (possibly empty) polygon in the image plane.

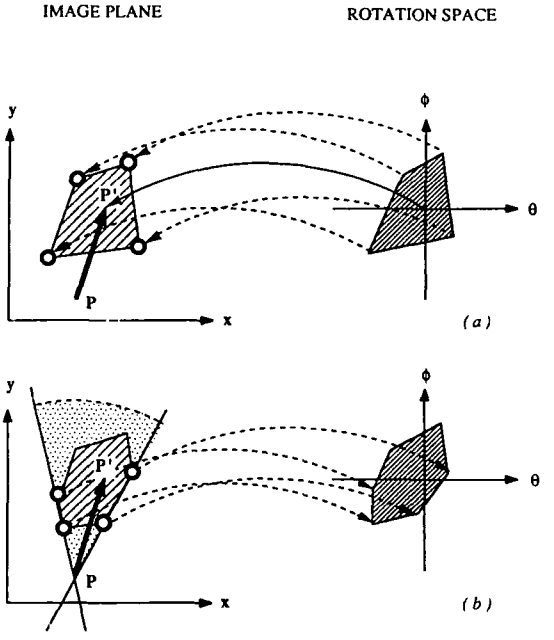



Figure 9: Successively constraining the range of possible camera rotations. *(a)* The rotation mappings corresponding to the vertices of the current rotation polygon are applied to every displacement vector ($\mathbf{P} \rightarrow \mathbf{P}'$). This yields a similar but distorted polygon in the image plane. *(b)* The polygon in the image is intersected with the open triangle defined by the tangents to the maximal FOE-region. Rotations that would bring the endpoint of the displacement vector outside this triangle are not feasible. The new vertices on the polygon are mapped back into rotation space.

*(d)* Map the new polygon from the image plane back into the rotation space (Figure 9(b)).

*(e)* If the rotation polygon is empty (number of vertices is zero), then stop. No camera rotation is possible that would make all displacement vectors intersect the given FOE-region. Repeat the process using a larger FOE-region.

Figures 10(a-c) show the changing shape of the rotation polygon during the application of this process to the three displacement vectors in Figure 8.

Since the mapping from rotation space to the image plane is nonlinear (equation 9), the straight lines between vertices in the rotation polygon do not correspond to straight lines in the image. They are, however, approximated as straight lines in order to simplify the intersection with the open triangle. The dotted lines in the image plane show the *actual* mapping of the rotation polygon onto the image. It can be seen that the deviations from straight lines are small and can be neglected.

Figure 10(c) shows the final rotation polygon after examining the three displacement vectors. The amount of actual camera rotation ($\theta$=–2.0°,$\phi$=5.0°) is marked with a small circle (arrow).

Of course, increasing the number of displacement vectors improves the rotation estimate. In practice, the amount of camera rotation can be constrained to a range of below 1° in both directions. It is interesting, although not surprising, that rotation can be estimated more accurately when the displacement vectors are short, i.e., when the amount of camera translation is small. This is in contrast to estimating camera translation which is easier with long displacement vectors.

The situation when the rotation polygon becomes *empty* requires some additional considerations. As mentioned earlier, in such a case no camera rotation is possible that would make all displacement vectors pass through the given FOE-region. This could indicate one of the two alternatives:

- At least one of the displacement vectors belongs to a moving object.
- The given FOE-region does not contain the actual location of the FOE, i.e., the region is *not feasible*.

The latter case is of particular importance. If a region can be determined *not* to contain the FOE, then the FOE must necessarily lie outside this region. Therefore, the above method can not only be used to estimate the amount of camera rotation, but also to search for the location of the FOE. Unfortunately, if the rotation polygon does not become empty, this does *not* imply that the FOE is actually inside the given region. It only means that all displacement vectors would *pass through* this region, not that they have a common *intersection* inside this region. However, if not all vectors pass through a certain region, then this region cannot possibly contain the FOE. The following recursive algorithm searches a given region for the FOE by splitting it into smaller pieces (divide-and-conquer):

```
MIN-FEASIBLE (region, min-size, disp-vectors):
  if SIZE (region) < min-size then return (region)
  else
    if FEASIBLE (region, disp-vectors) then
      return (union (
        MIN-FEASIBLE (sub-region-1, min-size,
              disp-vectors),
        MIN-FEASIBLE (sub-region-2, min-size,
              disp-vectors),
        ....
        MIN-FEASIBLE (sub-region-n, min-size,
              disp-vectors)))
    else return (nil) {region does not contain the FOE}
```

This algorithm searches for the smallest feasible FOE-region by systematically discarding subregions from further consideration. For the case that the shape of the original region is a square, subregions can be obtained by splitting the region into four subsquares of equal size.

The simple version shown here performs a depth-first search down to the smallest subregion (limited by the parameter "min-size"), which is neither the most elegant nor the most efficient approach. The algorithm can be significantly improved by applying a more sophisticated strategy, for example, by trying to discard subregions around the perimeter first before examining the interior of a region.

Two major problems were encountered with this method. *First*, the algorithm is computationally expensive since the process of computing feasible rotations must be repeated for every subregion. *Second*, a small region is more likely to be discarded than a larger one. However, when the size of the region becomes too small, errors induced by noise, distortion, or point-tracking may prohibit displacement vectors from passing though a region which actually contains the FOE.

Although this algorithm is not employed in the further treatment, it suggests an interesting alternative which departs significantly from traditional FOE-algorithms. Its main attractiveness is that it is inherently region-oriented in contrast to most other techniques which search for a single FOE-location. For the purpose of estimating the amount of rotation, the method using points at far distance mentioned earlier is probably more practical. Two other alternatives for locating the FOE once the rotation components have been estimated are discussed in the following.

### *3.2.2 Locating the FOE in a partially derotated image*

After applying a particular derotation mapping to the displacement field, the question is how close the new displacement field to a *radial mapping*, where all vectors diverge from one image location. If the displacement field is really radial, then the image is completely derotated and only the components due to camera translation remain. Two different methods for measuring this property are discussed. One method uses the *Variance of Intersection* at imaginary horizontal and vertical lines. The second method computes the *Linear Correlation Coefficient* to measure how "radial" the displacement field is.

*A. Variance of Intersection.* Prazdny[12] suggests to estimate the disturbance of the displacement field by computing the variance of intersections of one displacement vector with all other vectors. If the intersections lie in a small neighborhood, then the variance is small, which indicates that the displacement field is almost radial.

The problem can be simplified by using an imaginary horizontal and vertical line instead, whose orientation is not affected by different camera rotations. Figure 11 shows 5 displacement vectors $\mathbf{P}_1 \rightarrow \mathbf{P}_1' \ldots \mathbf{P}_5 \rightarrow \mathbf{P}_5'$ intersecting a vertical line at $x$ at $\bar{y}_1 \ldots \bar{y}_5$. Moving the vertical line from $x$ towards $x_0$ will bring the points of intersection closer together and will thus result in a smaller variance. The point of intersection of a displacement vector $\mathbf{P}_i \rightarrow \mathbf{P}_i'$ with a vertical line at $x$ is given by

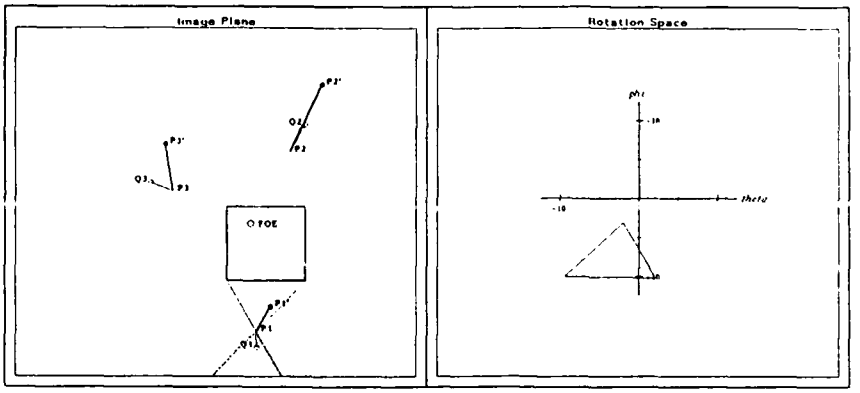



Figure 10: Changing rotation polygon. *(a)* The rotation polygon after examining displacement vector **P1→P1′**. Any camera rotation *inside* the polygon would move the endpoint of the displacement vector (**P1′**) into the open triangle formed by the tangents through **P1** to the maximal FOE-region given by the square in the image plane. The actual mapping of the rotation polygon into the image plane is shown with a dotted outline.

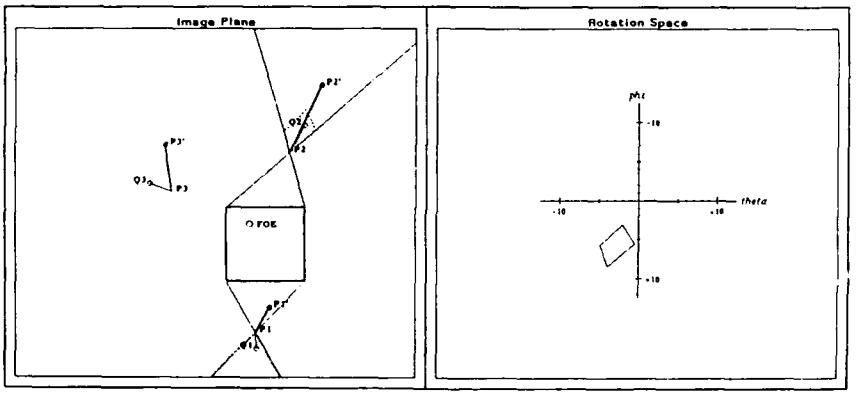



Figure 10: Changing rotation polygon. *(b)* The rotation polygon after examining displacement vectors **P1→P1′** and **P2→P2′**.

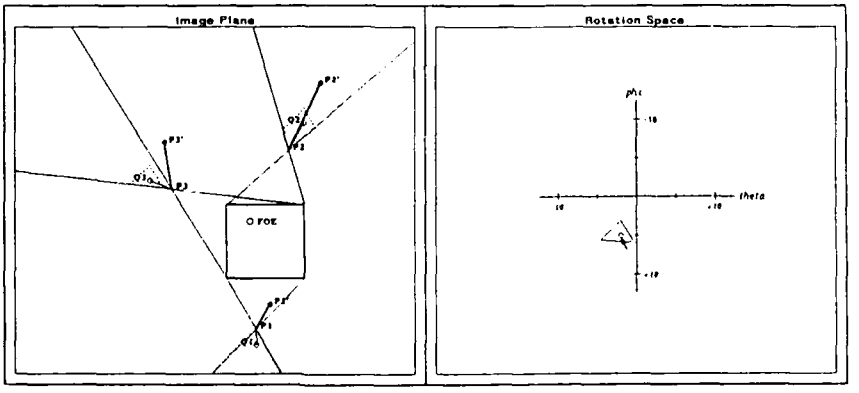



Figure 10: Changing rotation polygon. *(c)* The rotation polygon after examining displacement vector (**P1→P1′**, **P2→P2′**, and **P3→P3′**). **P1→P1'**, **P2→P2'** and **P3→P3'**. The amount of actual camera rotation is marked with a small circle (arrow).

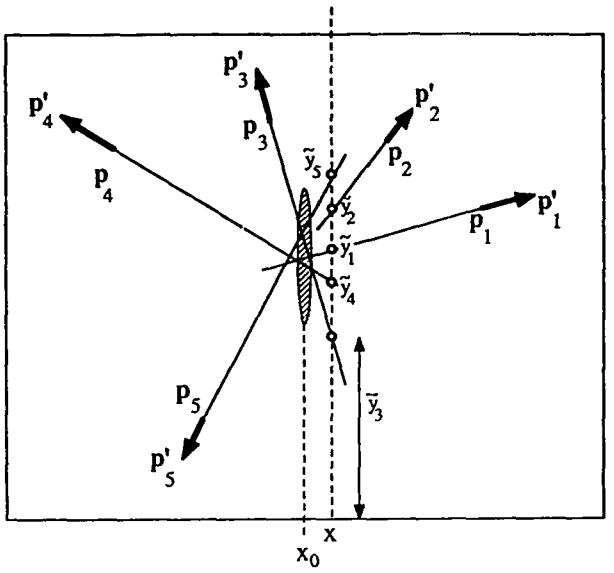



Figure 11: Intersecting the displacement vectors with a vertical line at $x$. When the vertical line is moved towards $x_0$ the points of intersection move closer together and therefore the *variance* of intersection decreases.

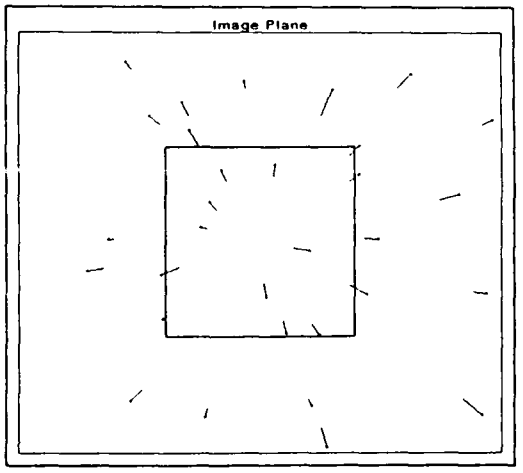



Figure 12: Displacement field used to evaluate various error functions. The square (±100 pixels in both directions) marks the region of evaluation.

$$\tilde{y}_i = \frac{x_i y_i' - y_i x_i'}{x_i - x_i'} \quad . \tag{28}$$

The variance of intersection of all displacement vectors with the vertical line at position $x$ is

$$\sigma^2(x) = \frac{1}{N}\left[\sum_{i,\, x_i \neq x_i'} \tilde{y}_i^2 - \frac{1}{N}\left[\sum_{i,\, x_i \neq x_i'} \tilde{y}_i\right]^2\right] . \tag{29}$$

To find the vertical cross section with minimum intersection variance, the first derivative of (29) with respect to $x$ is set to zero. The location $x_0$ of minimum intersection variance is then obtained.[1] Similarly, the position of a horizontal cross section with minimal intersection variance can be obtained.

The square root of the variance of intersection (standard deviation) at a vertical line was evaluated on the synthetic displacement field shown in Figure 12. The actual FOE is located in the center of the image. The square around the center (±100 pixels in both directions) marks the region over which the error functions are evaluated.

Figure 13 shows the distribution of the intersection standard deviation for increasing residual rotations in vertical direction in the absence of noise. Locations of displacement vectors are represented by real numbers (not rounded to integer values).

In Figure 13(a), no residual rotation exists, i.e., the displacement field is perfectly radial. The value of the horizontal position of the cross section varies ±100 pixels around the actual FOE. The standard deviation is zero for $x = x_f$ (the $x$-coordinate of the FOE) and increases linearly on both sides of the FOE. In Figures 13(b-d), the residual vertical rotation is increased from 0.2° to 1.0°. The bold vertical bar marks the horizontal position of minimum standard deviation, the thin bar marks the location of the FOE. It can be seen that the *amount* of minimum standard deviation rises with increasing disturbance by rotation, but that the *location* of minimum standard deviation does not necessarily move away from the FOE.

Figures 14-16 show the same function under the influence of noise. In Figure 14, noise was applied by merely rounding the locations of displacement vectors to their nearest integer values. Uniform noise of ±1 and ±2 pixels was added to image locations in Figures 15 and 16. It can be seen that the effects of noise are similar to the effects caused by residual rotation components. The purpose of this error function is to determine (a) where the FOE is located, and (b) how "radial" the current displacement field is.

If the displacement field is already perfectly derotated, then the location of minimum intersection standard deviation is, of course, the location of the FOE. Ideally all vectors pass through the FOE, such that a cross section through the FOE yields zero standard deviation. The question is how well the FOE can be located in an image which is *not* perfectly derotated.

Figure 17 plots the *location* of minimum intersection standard deviation under varying horizontal rotation. The vertical rotation is kept fixed for each plot. Horizontal camera rotations from -1° to +1° are shown on the abscissa (*rot*). The ordinate (*x0*) gives the location of minimum standard deviation in the range of ±100 pixels around the FOE (marked *xf*). It is not surprising that the location of minimum standard deviation depends strongly on the amount of horizontal rotation.

The problem is, however, that the location of minimum standard deviation is not necessarily closer to the FOE when

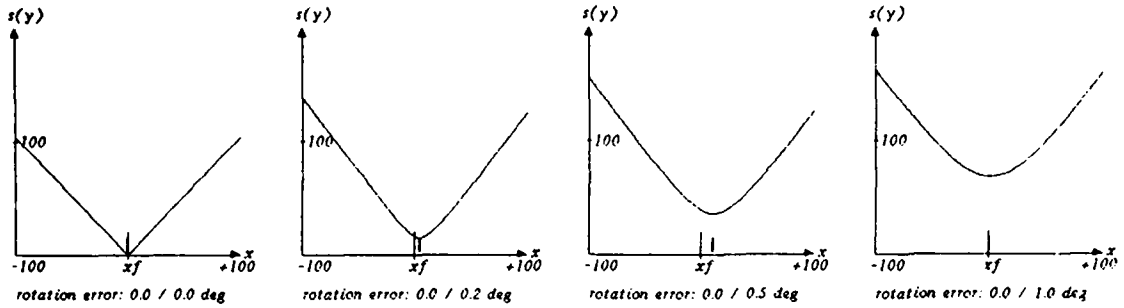



Figure 13: Standard deviation of intersection at a vertical cross section at position $x$ for different amounts of vertical rotation. *(a)* Without vertical rotation, *(b)* with 0.2° vertical rotation, *(c)* 0.5° and *(d)* 1.0°. The horizontal rotation is 0° in all cases. No noise was applied and image positions were not rounded to integers. The error values are shown for ±100 pixels around the $x$-coordinate of the FOE ($x_f$), which is marked with a thin bar. The location of minimum standard deviation is marked with a thick bar.

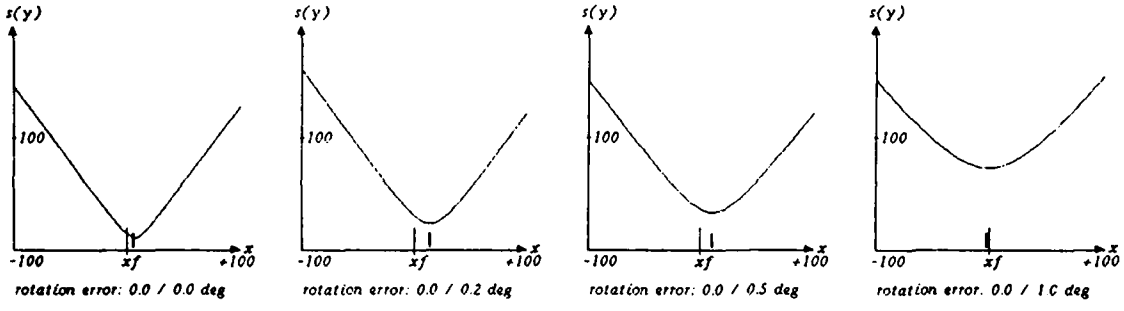



Figure 14: Standard deviation of intersection (square root) at a vertical cross section at position $x$ for different amounts of vertical rotation with no horizontal rotation. *(a)* Without vertical rotation, *(b)* with 0.2° vertical rotation, *(c)* 0.5°, and *(d)* 1.0°. No noise was applied and image positions were rounded to the closest integer values.

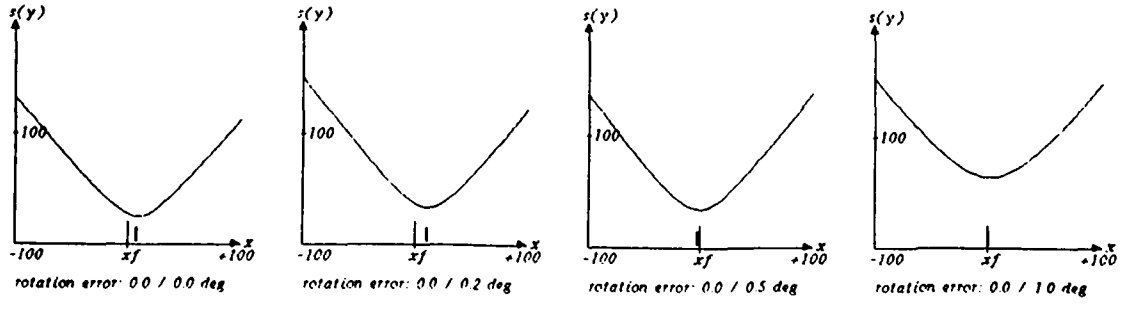



Figure 15: Standard deviation of intersection (square root) at a vertical cross section at position $x$ for different amounts of vertical rotation with no horizontal rotation. *(a)* Without vertical rotation, *(b)* with 0.2° vertical rotation, *(c)* 0.5°, and *(d)* 1.0°. Uniform noise of ±1 pixels was applied to the image locations.

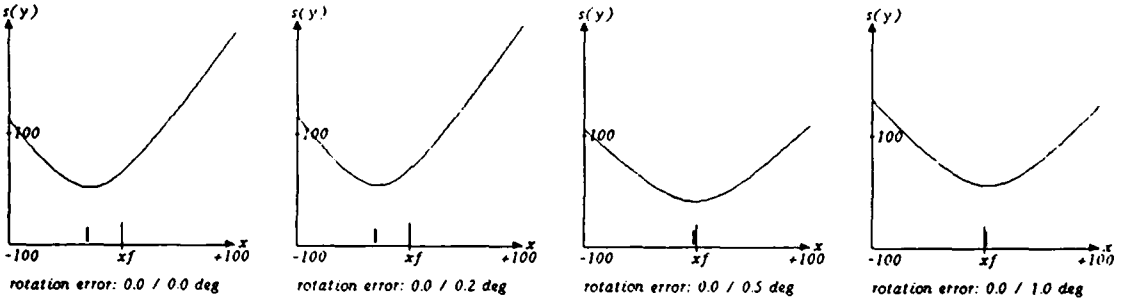



Figure 16: Standard deviation of intersection (square root) at a vertical cross section at position $x$ for different amounts of vertical rotation with no horizontal rotation. *(a)* Without vertical rotation, *(b)* with 0.2° vertical rotation, *(c)* 0.5°, and *(d)* 1.0°. Uniform noise of ±2 pixels was applied to the image locations.

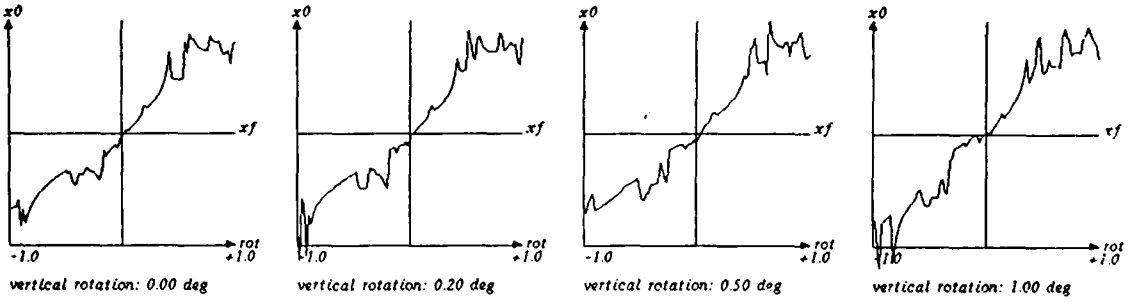



Figure 17: *Location* of minimum intersection standard deviation under varying horizontal rotation. The amount of vertical rotation is kept fixed in each plot. *(a)* Without vertical rotation, *(b)* with 0.2° vertical rotation, *(c)* 0.5°, and *(d)* 1.0°. Image locations were digitized but no noise was added. The horizontal location of the FOE is marked *xf*.

the amount of rotation is less. The function is only well behaved in a narrow range around zero rotation, which means that the estimate of the camera rotation must be very accurate to successfully locate the FOE.

The second purpose of this error function is to measure how "radial" the displacement field is after partial derotation. This should be possible by computing the *amount* of minimum intersection standard deviation. Intuitively, a smaller amount of minimum intersection standard deviation should indicate that the displacement field is less disturbed by rotation. Figure 18 and 19 show that this is generally true.

For the noise-free case in Figure 18, the amount of minimum intersection standard deviation becomes zero in the absence of horizontal and vertical rotations (a), indicating that the derotation is perfect. Unfortunately, the function is not well behaved even in this relatively small range of rotations (±1.0°). The curve exhibits some sharp local minima where an algorithm searching for an optimal derotation would get trapped easily. Figure 19 shows the same function in the presence of noise.

*B. Linear Correlation.* The second method of measuring how close a displacement field is to a radial pattern again uses the points of intersection at vertical (or horizontal) lines. The basic idea is illustrated in Figure 20. The displacement vectors are intersected by *two* vertical lines, both of which lie on the same side of the FOE. Since the location of the FOE is not known, the two lines are simply located at a sufficient distance from any possible FOE-location. This results in two sets of intersection points $\{(x_1,y_{1i})\}$ and $\{(x_2,y_{2i})\}$.

If all displacement vectors emanate from one single image location, then the distances between corresponding intersection points in the two sets must be proportional, i.e.,

$$\frac{y_{1i} - y_{1j}}{y_{2i} - y_{2j}} = \frac{y_{1j} - y_{1k}}{y_{2j} - y_{2k}} \quad \text{for all } i,j,k\,. \tag{30}$$

Therefore, a linear relationship exists between the vertical coordinates of intersection points on these two lines. The "goodness" of this linear relationship is easily measured by computing the *correlation coefficient* for the $y$-coordinates of the two sets of points.[1]

The resulting coefficient is a real number in the range from -1.0 to +1.0. If both vertical lines are on the same side of the FOE, then the optimal value is +1.0. Otherwise, if the FOE lies between the two lines, the optimal coefficient is -1.0. The horizontal position of the two vertical lines is of no importance, as long as one of these conditions is satisfied. For example, the left and right border lines of the image can be used.

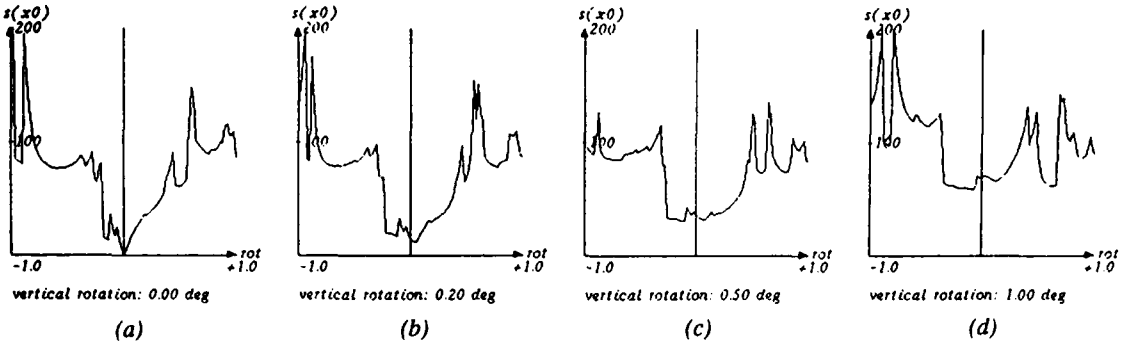



Figure 18: *Amount* of minimum intersection standard deviation under varying horizontal rotation. The amount of vertical rotation is kept fixed in each plot. *(a)* Without vertical rotation, *(b)* with 0.2° vertical rotation, *(c)* 0.5°, and *(d)* 1.0°. Image locations were digitized but no noise was added.

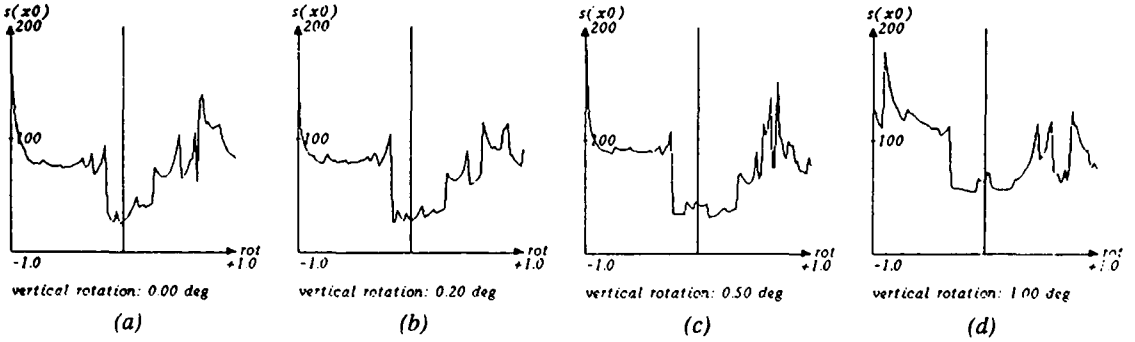



Figure 19: *Amount* of minimum intersection standard deviation under varying horizontal rotation as in Figure 18. *(a)* Without vertical rotation, *(b)* with 0.2° vertical rotation, *(c)* 0.5°, and *(d)* 1.0°. Uniform noise of ±2 pixels was added to image locations.

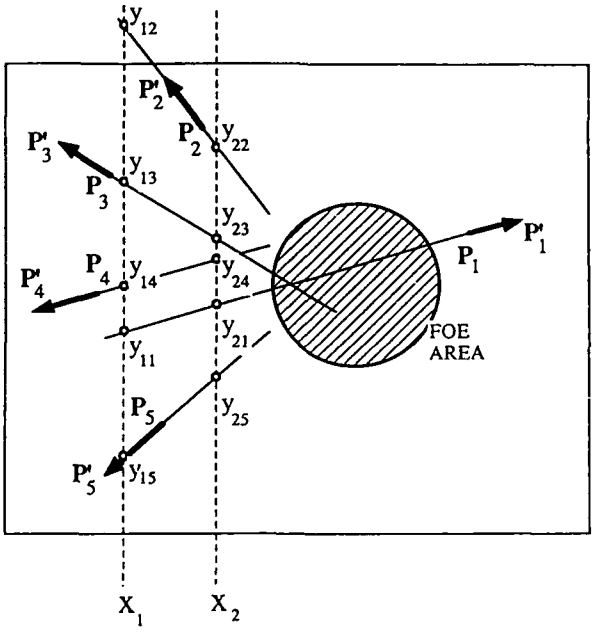
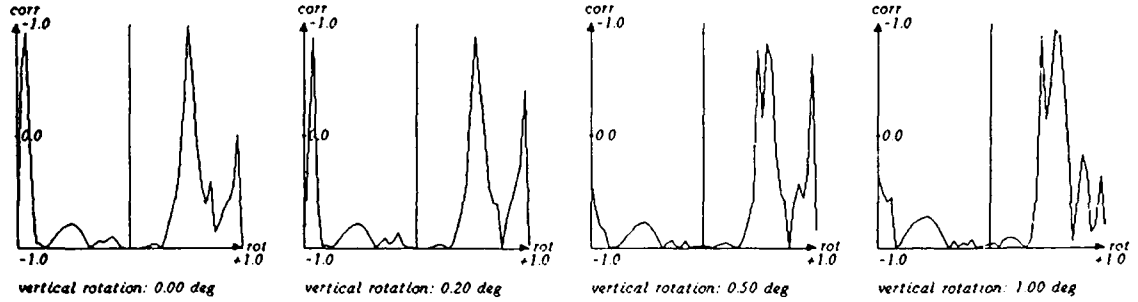
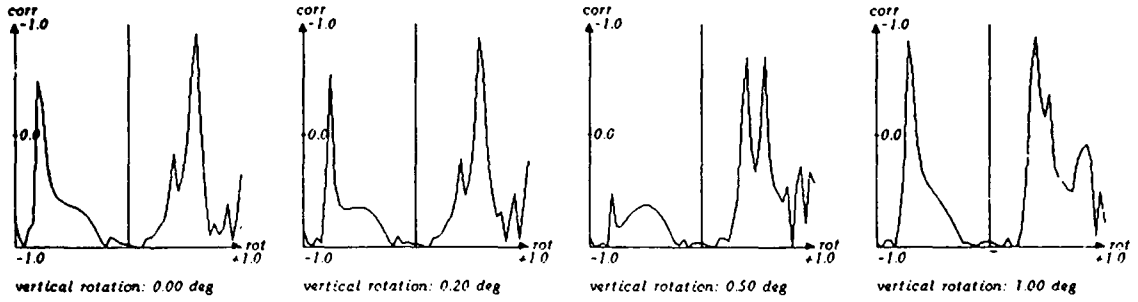



Figure 20: Intersecting displacement vectors with two vertical lines, both of which lie on the same side of the FOE.




*(a)* *(b)* *(c)* *(d)*

Figure 21 Correlation coefficient for the intersection of displacement vectors at two vertical lines under varying horizontal rotations in the noise-free case. *(a)* Without vertical rotation, *(b)* with 0.2° vertical rotation, *(c)* 0.5°, and *(d)* 1.0°. The optimal coefficient is +1.0 (horizontal axis).




*(a)* *(b)* *(c)* *(d)*

Figure 22 Correlation coefficient for the intersection of displacement vectors at two vertical lines under varying horizontal rotations. Uniform noise of ±2 pixels was added to image locations. Without vertical rotation *(a)*, with 0.2° vertical rotation *(b)*, 0.5° *(c)* and 1.0° *(d)*. The optimal coefficient is +1.0 (horizontal axis).

Figures 21 and 22 show plots for this correlation coefficient under the same conditions as in Figures 18 and 19. No noise was applied for Figure 21. The shapes of the curves are similar to those for the minimum standard deviations shown earlier, with peaks at the same locations. It is apparent, however, that each curve has several locations where the coefficient is close to the optimum value (+1.0), i.e., no distinct global optimum exists which is not only the case in the presence of noise (Figure 22). This fact makes the method of maximizing the correlation coefficient useless for computing the FOE.

### 3.3 Rotation from FOE

The main problem encountered in computing the FOE in section 3.2 was that none of the functions examined was well behaved, making the search for an optimal derotation and the location of the FOE difficult. Disturbances induced by noise and residual rotation components are amplified by extending short displacement to straight lines and computing their intersections. The method described below avoids this problem by guessing an FOE-location first and estimating the optimal derotation for this particular FOE in the second step.

Given the two images $\mathbf{I}_0$ and $\mathbf{I}_1$ of corresponding points, the main algorithmic steps of this approach are:

*(1)* Guess an FOE-location $\mathbf{x}_f^{(i)}$ in image $\mathbf{I}_0$ (for the current iteration $i$).

*(2)* Determine the derotation mapping $\mathbf{r}_\theta^{-1}$, $\mathbf{r}_\phi^{-1}$ which would transform image $\mathbf{I}_1$ into an image $\mathbf{I}_1'$ such that the mapping $(\mathbf{x}_f^{(i)}, \mathbf{I}_0, \mathbf{I}_1')$ deviates from a radial mapping (23) with minimum error $E^{(i)}$.

*(3)* Repeat steps *(1)* and *(2)* until an FOE-location $\mathbf{x}_f^{(k)}$ with the lowest minimum error $E^{(k)}$ is found.

An *initial guess* for the FOE-location is obtained from knowledge about the orientation of the camera with respect to the vehicle. For subsequent pairs of frames, the FOE-location computed from the previous pair can be used as a starting point.

Once a particular $\mathbf{x}_f$ has been selected, the problem is to compute the rotation mappings $\mathbf{r}_\theta^{-1}$ and $\mathbf{r}_\phi^{-1}$ which, when applied to the image $\mathbf{I}_1$, will result in an optimal radial mapping with respect to $\mathbf{I}_0$ and $\mathbf{x}_f$.

To measure how close a given mapping is to a radial mapping, the perpendicular distances between points in the second image ($\mathbf{x}_i'$) and the "ideal" displacement vectors is measured. The "ideal" displacement vectors lie on straight lines passing through the the FOE $\mathbf{x}_f$ and the points in the first image $\mathbf{x}_i$ (Figure 23). The sum of the squared perpendicular distances $d_i$ is the final error measure. For each set of corresponding image points ($\mathbf{x}_i \in \mathbf{I}$, $\mathbf{x}_i' \in \mathbf{I}'$), the error measure is defined as

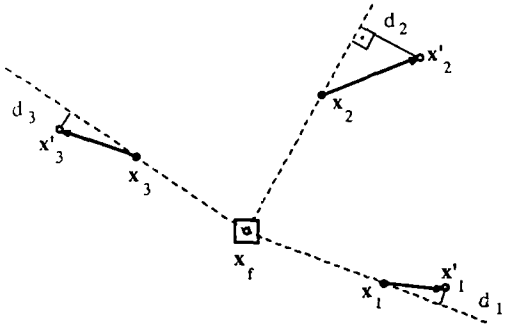



Figure 23: Measuring the perpendicular distance $d_i$ between lines from $\mathbf{x}_f$ through points $\mathbf{x}_i$ and points $\mathbf{x}_i'$ in the second image.

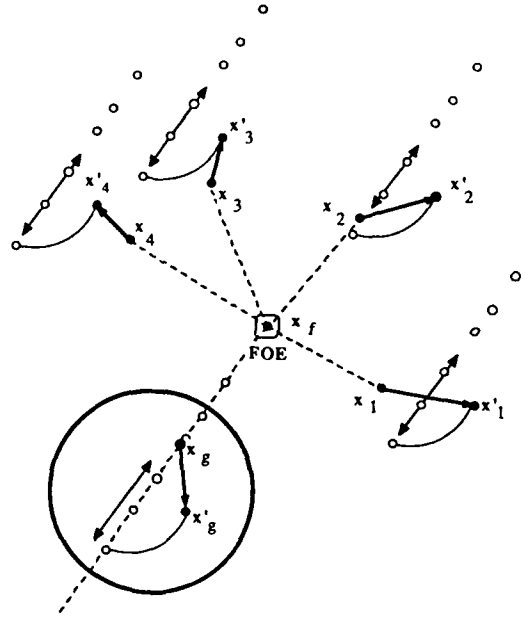



Figure 24: One vector $\mathbf{x}_g$ is selected from the set of displacement vectors to determine the optimum 2-D shift to be applied to points $\mathbf{x}_i'$, given a FOE-location $\mathbf{x}_f$. First $\mathbf{x}_g'$ is forced onto the line $\overline{\mathbf{x}_f\mathbf{x}_g}$ and then the entire image $\mathbf{I}' = \{\mathbf{x}_1', \mathbf{x}_2', \ldots\}$ is translated in the direction of this line until the error value reaches a minimum.

$$E(\mathbf{x}_f) = \sum_i E_i = \sum_i d_i^2 = \sum_i \left[ \frac{1}{|\vec{\mathbf{x}_f\mathbf{x}_i}|} \vec{\mathbf{x}_f\mathbf{x}_i} \times \vec{\mathbf{x}_f\mathbf{x}_i'} \right]^2 . \quad (31)$$

In the following, it is assumed that the amount of residual image rotation in horizontal and vertical direction is moderately small (less than 4°). In most practical cases, this condition is satisfied, provided that the time interval between frames is sufficiently small. However, should the amount of vehicle rotation be very large for some reason, a coarse estimate of the actual rotation can be found (as described earlier) and applied to the image before the FOE computation. With small amounts of rotation, the actual rotation mapping, where points move on horizontal and vertical hyperbolic paths, can be approximated by a horizontal and vertical *shift* with constant length over the entire image.

Under this condition, the inverse rotation mapping $\mathbf{r}_\phi^{-1}$, $\mathbf{r}_\theta^{-1}$ can be approximated by a adding a constant vector $\mathbf{s} = (s_x\ s_y)$ which is independent of the image location:

$$\mathbf{I}_1' = \mathbf{r}_\theta^{-1}\mathbf{r}_\phi^{-1}\,\mathbf{I}_1 \approx \mathbf{s} + \mathbf{I}_1 . \quad (32)$$

Given two images $\mathbf{I}$ and $\mathbf{I}'$ the error measure (31) becomes

$$E(\mathbf{x}_f, \mathbf{s}) = \sum_i \left\{ \frac{1}{|\vec{\mathbf{x}_f\mathbf{x}_i'}|^2} \left[ \vec{\mathbf{x}_f\mathbf{x}_i} \times (\vec{\mathbf{x}_f\mathbf{x}_i'} + \mathbf{s}) \right]^2 \right\} \quad (33)$$

where $\mathbf{x}_i \in \mathbf{I}$ and $\mathbf{x}_i' \in \mathbf{I}'$. For a given FOE-location $\mathbf{x}_f$, the problem is to minimize E with respect to the two unknowns $s_x$ and $s_y$. To reduce this problem to a one-dimensional search, one point $\mathbf{x}_g$, called the *Guiding Point*, is selected in image $\mathbf{I}$ which is forced to maintain zero error (Figure 24). Therefore, the corresponding point $\mathbf{x}_g'$ must lie on a straight line passing through $\mathbf{x}_f$ and $\mathbf{x}_g$ Any shift s applied to the image $\mathbf{I}'$ must keep $\mathbf{x}_g'$ on this straight line, so

$$\mathbf{x}_g' + \mathbf{s} = \mathbf{x}_f + \lambda\,(\mathbf{x}_g - \mathbf{x}_f) \quad \text{for all s,} \quad (34a)$$

and thus,

$$\mathbf{s} = \mathbf{x}_f - \mathbf{x}_g' + \lambda\,(\mathbf{x}_g - \mathbf{x}_f) \quad (\lambda \in R). \quad (34b)$$

For $\lambda = 1$, $\mathbf{s} = \mathbf{x}_g - \mathbf{x}_g'$ which is the vector $\mathbf{x}_g' \rightarrow \mathbf{x}_g$. This means that the image $\mathbf{I}'$ is shifted such that $\mathbf{x}_g$ and $\mathbf{x}_g'$ overlap. This leaves $\lambda$ as the only free variable and the error function (33) is obtained as

$$E(\lambda) = \sum_i \left[ \lambda\, A_i + B_i - C_i \right]^2 \quad (35)$$

with

$$l_{if} = \sqrt{(x_i - x_f)^2 + (y_i - y_f)^2}$$

$$A_i = \frac{1}{l_{if}} (y_i - y_f)(x_g - x_f) - (x_i - x_f)(y_g - y_f)$$

$$B_i = \frac{1}{l_{if}} (y_i - y_f)(x_i' - x_g')$$

$$C_i = \frac{1}{l_{if}} (x_i - x_f)(y_i' - y_g') .$$

Differentiating 35 with respect to $\lambda$ and forcing the resulting equation to zero yields the parameter for the optimal shift $\mathbf{s}_{opt}$ as

$$\lambda_{opt} = \frac{\sum A_i C_i - \sum A_i B_i}{\sum A_i^2} . \quad (36)$$

The optimal shift $\mathbf{s}_{opt}$ and the resulting minimum error $E(\lambda_{opt})$ for the given FOE-location $\mathbf{x}_f$ is obtained by inserting $\lambda_{opt}$ into equations (34b) and (35) respectively, giving

$$E_{min}(\mathbf{x}_f) = \lambda_{opt}^2 \sum A_i^2 + 2\,\lambda_{opt} \left[ \sum A_i B_i - \sum A_i C_i \right] \quad (37)$$
$$- 2 \sum B_i C_i + \sum B_i^2 + \sum C_i^2 .$$

The normalized error $E_n$ shown in the following results (Figures 26-31) is defined as

$$E_n(\mathbf{x}_f) = \sqrt{\frac{1}{N} E_{min}(\mathbf{x}_f)} \quad (38)$$

where $N$ is the number of displacement vectors used for com-

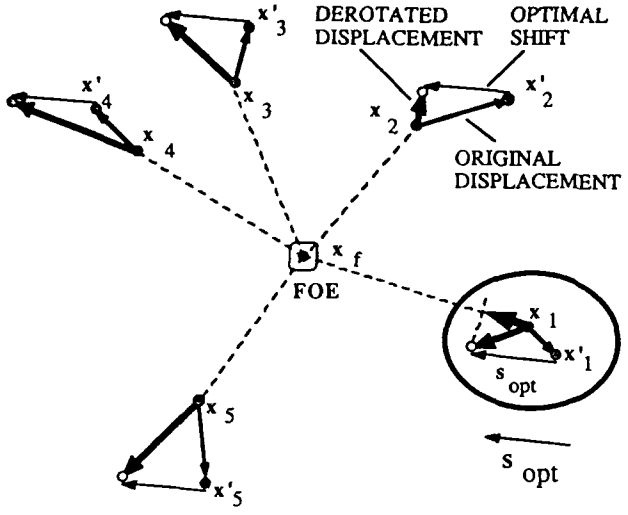



Figure 25: FOE-locations are *prohibited* if the displacement field resulting from the application of the optimal shift $s_{opt}$ contains vectors pointing towards the FOE. This is the case at point $x_1$.

puting the FOE.

Since in a displacement field caused by pure camera translation all vectors must point away from the FOE, this restriction must hold for any candidate FOE-location (Figure 24). If after applying $s_{opt}(x_f)$ to the second image I', the resulting displacement field contains vectors pointing *towards* the hypothesized $x_f$, then this FOE-location is *prohibited* and can be discarded from further consideration. Figure 25 shows a field of 5 displacement vectors. The optimal shift $s_{opt}$ for the given $x_f$ is shown as a vector in the lower right-hand corner. When $s_{opt}$ is applied to point $x_1'$, the resulting displacement vector (shown fat) does not point away from the FOE. Since its projection onto the line $x_f x_1$ points *towards* the FOE, it is certainly not consistent with a radial expansion pattern.

The final algorithm for determining the direction of heading as well as horizontal and vertical camera rotations is the following:

Find-FOE:

*(1)* Guess an initial FOE $x_f^0$, for example the FOE-location obtained from the previous pair of frames.

*(2)* Starting from $x_f^0$, search for a location $x_f^{opt}$ where $E_{min}(x_f^{opt})$ is a minimum. A technique of *steepest descent* is used, where the search proceeds in the direction of least error.

*(3)* Determine a region around $x_f^{opt}$ in which the error is below some threshold.

The search for this FOE-*area* is conducted at FOE-locations lying on a grid of fixed width. In the examples shown, the grid spacing is 10 pixels on both *x*- and *y*-directions.

The error function $E(x_f)$ is computed in time proportional to the number of displacement vectors *N*. The final size of the FOE-area depends on the local shape of the error function and can be constrained not to exceed a certain maximum *M*. Therefore, the time complexity is $O(MN)$.

### 3.4 Experiments on Synthetic Data

The first set of experiments was conducted on synthetic imagery to investigate the behavior of the error measure under various conditions, namely

- the average length of the displacement vectors (longer displacement vectors lead to a more accurate estimate of the FOE),
- the amount of residual rotation components in the image, and
- the amount of noise applied to the location of image points.

Figure 26 shows the distribution of the normalized error $E_n(x_f)$ for a sparse and relatively short displacement field containing 7 vectors. Residual rotation components of ±2° in horizontal and vertical direction are present in *(b)-(d)* to visualize their effects upon the image. This displacement field was used with different average vector lengths (indicated as *length-factor*) for the other experiments on synthetic data. The displacement vector through the *Guiding Point* is marked with a heavy line. The choice of this point is not critical, but it should be located at a considerable distance from the FOE to reduce the effects of noise upon the direction of the vector $\vec{x_f x_g}$.

In Figure 26, the error function is sampled in a grid with a width of 10 pixels over an area of 200 by 200 pixels around the actual FOE, which is marked by a small square. At each grid point, the amount of error is indicated by the size of the circle. Heavy circles indicate error values which are above a certain threshold. Those FOE-locations that would result in displacement vectors which point *towards* the FOE (as described earlier) are marked as prohibited (+). It can be seen that the shape of the 2D error function changes smoothly with different residual rotations over a wide area and exhibits its minimum close to the actual location of the FOE.

Figures 27 to 32 show the effects of various conditions upon the behavior of this error function in the same 200×200 pixel square around the actual FOE as in Figure 26.

Figure 27 shows how the shape of the error function depends upon the average length of the displacement vectors in the absence of any residual rotation or noise (except digitization noise). Clearly, the minimum of the error function becomes more distinct with increasing amounts of displacement.

Figure 28 shows the effect of increasing residual rotation in horizontal direction upon the shape of the error function.

Figure 29 shows the effect of residual rotation in vertical direction. Here, it is important to notice that the displacement field used is extremely nonsymmetric along the Y-axis of the image plane. This is motivated by the fact that in real AUV images, long displacement vectors are most likely to be found from points on the ground, which are located in the lower portion of the image. Therefore, positive and negative vertical rotations have been applied in Figure 29.

In Figure 30, residual rotations in both horizontal and vertical direction are present. It can be seen (Figure 30*(a-e)*) that the error function is quite robust against rotational components in the image. Figure 30*(f-j)* shows the amounts of optimal linear shift $s_{opt}$ under the same conditions.

The result in Figure 30*(e)* shows the effect of large combined rotation of 4.0° / 4.0° in both directions. Here, the minimum of the error function is considerably off the actual

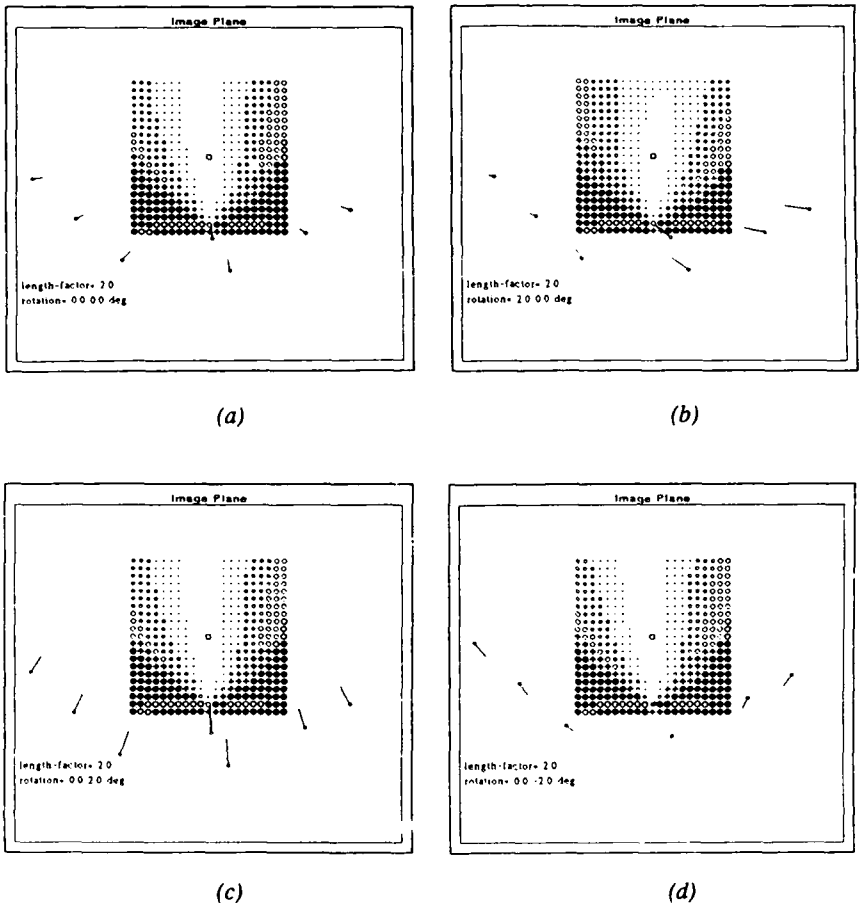



Figure 26: Displacement field and minimum error at selected FOE-locations. The shape of the error function is plotted over an area of ±100 pixels around the actual FOE, which is marked with a small square. The diameter of the circle drawn at each hypothesized FOE-location indicates the amount of normalized error (equation 40), large circles are locations of large error. Heavy circles indicate error values above a certain threshold (4.0), *prohibited* locations (as defined earlier) are marked "+". *(a)* No residual rotation. *(b)* 2.0° of horizontal camera rotation (camera rotated to the left). *(c)* 2.0° vertical rotation (camera rotated upwards). *(d)* -2.0° vertical rotation (camera rotated downwards).

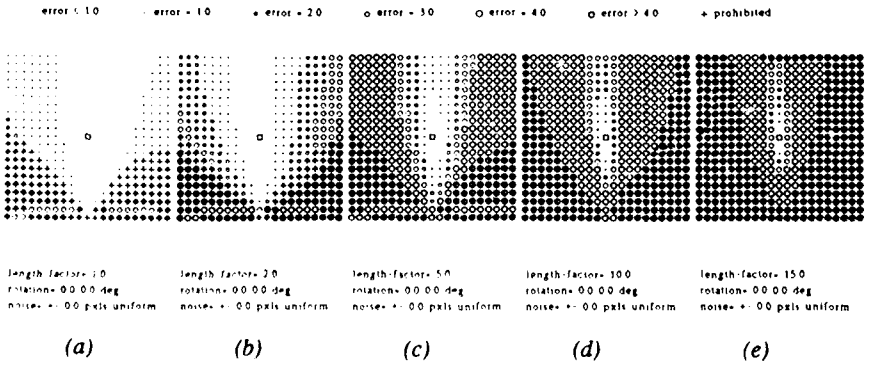



Figure 27 (a-e): Effects of increasing the average length of displacement vectors upon the shape of the error function. Length factors vary from 1 to 15. The error function was evaluated over the same image area of 200×200 pixels around the actual FOE (square) as in Figure 26. No rotation or noise was applied.

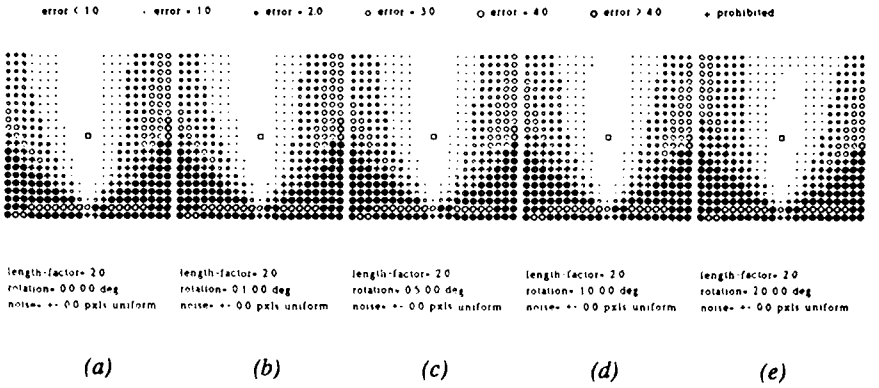



Figure 28 (a-e): Effects of increasing residual rotation in horizontal direction upon the shape of the error function for relatively short vectors (length-factor 2.0). No noise was applied.

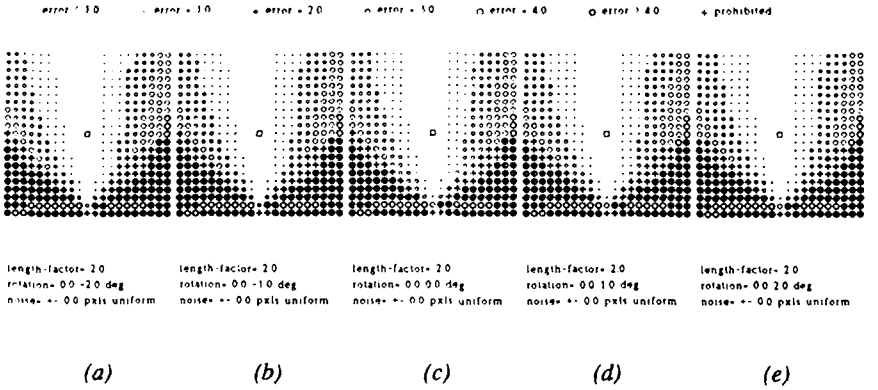


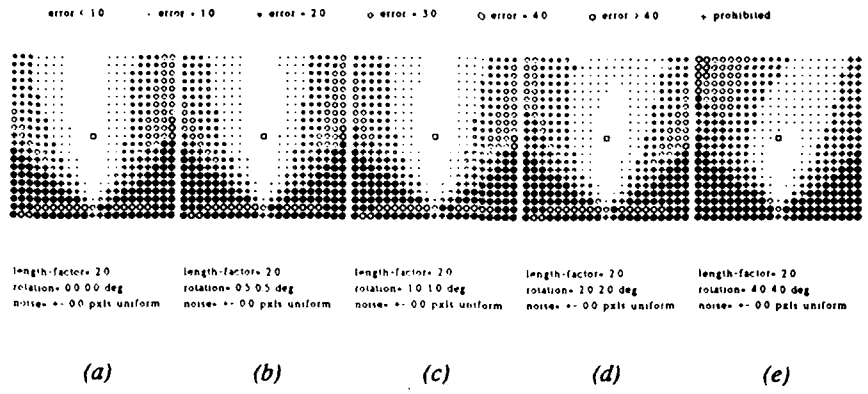

Figure 29 (a-e): Effects of increasing residual rotation in vertical direction upon the shape of the error function for relatively short vectors (length-factor 2.0). No noise was applied.

location of the FOE because of the error induced by using a linear shift to approximate the nonlinear derotation mapping. In such a case, it would be necessary to actually *derotate* the displacement field by the amount of rotation equivalent to $s_{opt}$ found at the minimum of this error function and repeat the process with the derotated displacement.

The effects of various amounts of noise are shown in Figure 31. For this purpose, a random amount (with uniform distribution) of displacement was added to the original (continuous) image location and then rounded to integer pixel coordinates. Random displacement was applied in ranges from ±0.5 to ±4.0 pixels in both horizontal and vertical direction. Since the displacement field contains only 7 vectors, the results do not provide information about the statistical effects of image noise. This would require more extensive modeling and simulation. However, what can be observed here is that the absolute minimum error increases with the amount of noise. It can thus serve as an indicator for the amount of noise present in the image and the reliability of the final result.

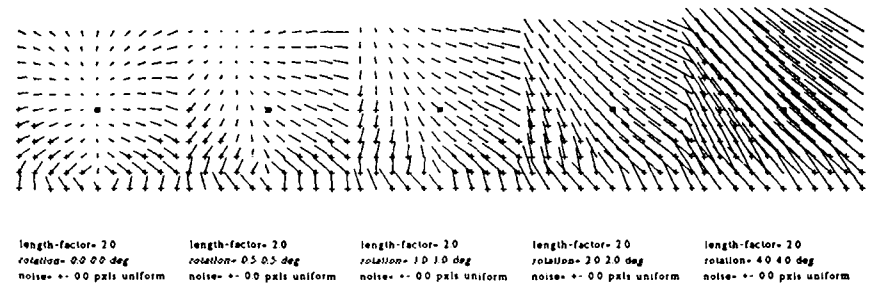
Again, the length of the displacement vectors is an important factor. The shorter the displacement vectors are, the more difficult it is to locate the FOE correctly in the presence of noise. Figure 32 shows the error functions for two displacement fields with different average vector lengths. For the shorter displacement field (*length-factor* 2.0) in Figure 32*(a)*, the shape of the error function changes dramatically under the same amount of noise (compare Figure 30*(a)*). A search for the minimum error would inevitably converge towards a point indicated by the small arrow, far off the actual FOE. For the image with *length-factor* 5.0 (Figure 32*(b)*), the minimum of the error function coincides with the actual location of the FOE *(a)*. The different result for the same constellation of points in the Figure 31*(d)* is caused by the different random numbers (noise) obtained in each experiment. This experiment shows that a sufficient amount of displacement between consecutive frames is essential for reliably determining the FOE and thus, the direction of vehicle translation.

The performance of this FOE algorithm is shown in section 5.1 on a sequence of real images taken from the moving ALV. In the following section, it is shown how the absolute velocity of the vehicle can be estimated after the location of the FOE has been determined. The essential measure used for this calculation is the absolute height of the camera above the ground which is constant and known. Given the absolute velocity of the vehicle, the absolute *distance* from the camera of 3D points in the scene can be estimated using equation (20).



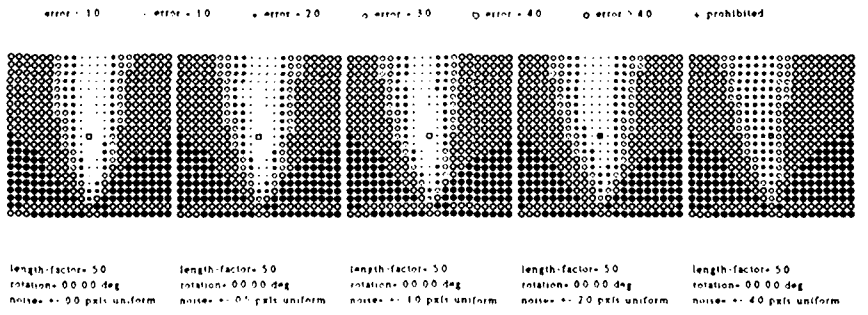

*(a)* *(b)* *(c)* *(d)* *(e)*

Figure 30 (a-e): Effects of increasing residual rotation in horizontal and vertical direction upon the shape of the error function for relatively short vectors (length-factor 2.0). No noise was applied.



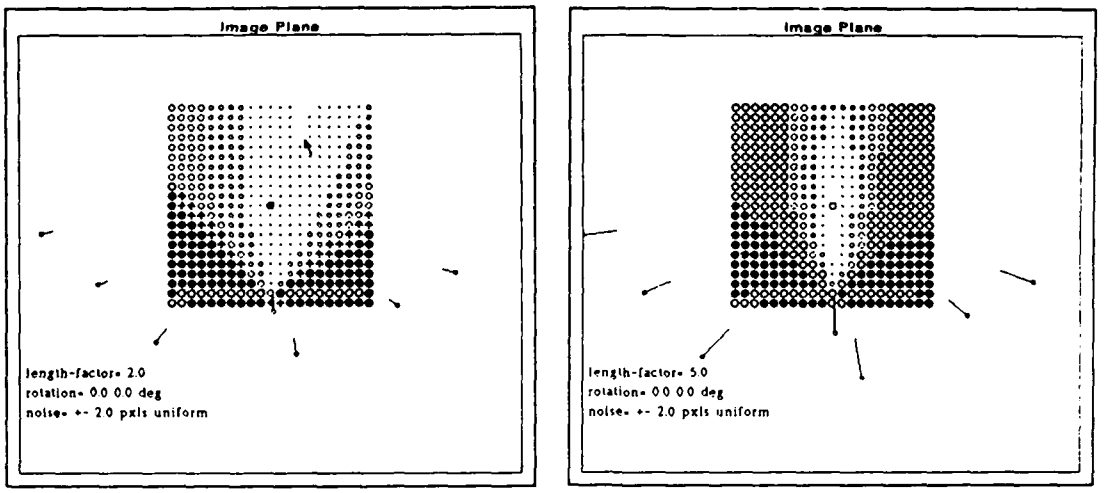

*(f)* *(g)* *(h)* *(i)* *(j)*

Figure 30 (f-j): The amount of optimal linear shift obtained under the same conditions as in Figure 30 *(a-e)*.




*(a)* *(b)* *(c)* *(d)* *(e)*

Figure 31 (a-e): The effects of uniform noise applied to image point coordinates for a constant average vector length. The shape of the error function become flat around the local minimum of the FOE with increasing levels of noise.




*(a)* *(b)*

Figure 32: The effects of uniform noise applied to image point coordinates for different average vector lengths (length factors 2.0 and 5.0). For the short displacement field *(a)* the disturbance moves the local minimum (arrow) far off the actual FOE. The same amount of noise applied to the longer displacement field has much less dramatic effects.

### 3.4 Computing Velocity Over Ground

After the FOE has been computed following the steps outlined in the previous section, the direction of vehicle translation and the amount of rotation are known. From the derotated displacement field and the location of the FOE, the 3D layout of the scene can be obtained up to a common scale factor (20). As pointed out earlier, this scale factor and, consequently, the velocity of the vehicle can be determined if the 3D position of one point in space is known. Furthermore, it is easy to show[6,4] that it is sufficient to know only one coordinate value of a point in space to reconstruct its position in space from its location in the image.

Since the ALV travels on a fairly flat surface, the road can be approximated as a plane which lies parallel to the vehicle's direction of translation (see Figure 33). This approximation holds at least for a good part of the road in the field of view of the camera.

Since the absolute height of the camera above the ground is constant and known, it should be possible to estimate the positions of points on the road surface with respect to the vehicle in *absolute* terms. From the changing distances between these points and the camera, the actual advancement and speed can be determined.

First, a new coordinate system is introduced which has its origin in the lens center of the camera. The Z-axis of the new system passes through the FOE in the image plane and points, therefore, in the direction of translation. The original camera-centered coordinate system (X Y Z) is transformed into the new frame (X' Y' Z') merely by applying horizontal and vertical rotation until the Z-axis lines-up with the FOE.

The horizontal and vertical orientation in terms of *pan* and *tilt* are obtained by "rotating" the FOE $(x_f\, y_f)$ into the center of the image (0 0) using equations (14) and (15):

$$\theta_f = -\tan^{-1}\frac{x_f}{f} \tag{39}$$

$$\phi_f = -\tan^{-1}\left[y_f \frac{f^2}{(f^2 + x_f^2)\, f^2 - x_f^2\, y_f^2}\right]. \tag{40}$$

The two angles $\theta_f$ and $\phi_f$ represent the orientation of the camera in 3D with respect to the new coordinate system. This allows us to determine the 3D orientation of the projecting rays passing through image points by use of the inverse perspective transformation. A 3D point $\mathbf{X}$ in the environment whose image $\mathbf{x} = (x\ y)$ is given, lies on a straight line in space defined by

$$\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \kappa \begin{bmatrix} \cos\theta_f & \sin\theta_f \sin\phi_f & -\sin\theta_f \cos\phi_f \\ 0 & \cos\phi_f & \sin\phi_f \\ \sin\theta_f & -\cos\theta_f \sin\phi_f & \cos\theta_f \cos\phi_f \end{bmatrix} \begin{bmatrix} x \\ y \\ f \end{bmatrix} \tag{41}$$

For points on the road surface, the $Y$-coordinate is $-h$ which is the height of the camera above ground. Therefore, the value of $\kappa_s$ for a point on the road surface $(x_s\, y_s)$ can be estimated as

$$\kappa_s = \frac{-h}{y_s \cos\theta_f + f \sin\theta_f} \tag{42}$$

and its 3D distance is found by inserting $\mathbf{v}_s$ into equation 41 as

$$Z_s = -h\, \frac{x_s \sin\theta_f - y_s \cos\theta_f \sin\phi_f - f \cos\theta_f \cos\phi_f}{y_s \cos\phi_s + f \sin\phi_s}. \tag{43}$$

If a point on the ground is observed at two instances of time, $\mathbf{x}_s$ at time $t$ and $\mathbf{x}_s'$ at $t'$, the resulting distances from the vehicle $Z_s$ at $t$ and $Z_s'$ at $t'$ yield the amount of advancement

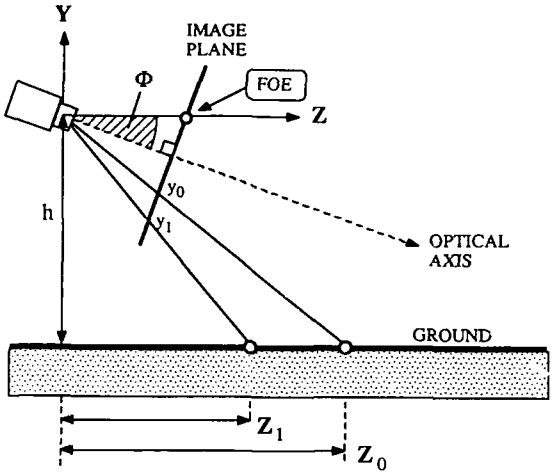



Figure 33: Side view of the camera traveling parallel to a flat surface. The camera advanced in direction **Z**, such that a 3D point on the ground moves relative to the camera from $\mathbf{Z}_0$ to $\mathbf{Z}_1$. The depression angle $\phi$ can be found from the location of the FOE in the image. The height of the camera above the ground is given.

$\Delta Z_s\ (t,t')$ and estimated velocity $V_s\ (t,t')$ in this period as

$$\Delta Z_s\ (t,t') = Z_s - Z_s' \tag{44}$$

$$V_s\ (t,t') = \frac{Z_s - Z_s'}{t' - t}\ . \tag{45}$$

Of course, image noise and tracking errors have a large impact upon the quality of the final velocity estimate. Therefore, the longest available displacement vectors are generally selected for this measurement, i.e., those vectors which are relatively close to the vehicle. Also, in violation of the initial assumption, the ground surface is never perfectly flat. In order to partially compensate these errors and to make the velocity estimate more reliable, the results of the measurements on individual vectors are combined. The length of each displacement vector $|\mathbf{x}_i - \mathbf{x}_i'|$ in the image is used as the weight for its contribution to the final result. Given a set of suitable displacement vectors $S = \{\ \mathbf{x}_i - \mathbf{x}_i'\ \}$, the estimate of the distance traveled by the vehicle is taken as the weighed average of the measurements $\Delta Z_i$ on individual vectors

$$\tilde{\Delta Z}\ (t,t') = \frac{\sum\ (|\mathbf{x}_i - \mathbf{x}_i'|\ \Delta Z_i)}{\sum |\mathbf{x}_i - \mathbf{x}_i'|} \tag{46}$$

and the final estimate for the vehicle velocity is

$$\tilde{V}\ (t,t') = \frac{\tilde{\Delta Z}}{t' - t}\ . \tag{47}$$

This computation was applied to a sequence of real images which is described in section 5.

## 4. THE QUALITATIVE SCENE MODEL

The choice of a suitable scheme for the internal representation of the scene is of great importance. The *Qualitative Scene Model* (QSM) is a 3D camera-centered interpretation of the scene that is built incrementally from visual information gathered over time. The nature of this model, however, is *qualitative* rather than a precise geometric description of the scene. The basic building blocks of the QSM are *entities*, which are the 3D counterparts of the 2D *features* observed in the image. For example, the point feature *A* located in the image at *x,y* at time *t*

( Point_Feature *A t x y* )

has its 3D counterpart in the model as

( Point_Entity *A* ).

Since the model is camera-centered ("retinocentric"), the image locations and 2D movements of features are implicitly part (i.e., known facts) of the model. Additional entries are the properties of entities (e.g., "stationary" or "mobile") and relationships between entities (e.g. "closer"), which are not given facts but hypotheses about the real scene. This is expressed in the model as either

( Stationary *entity* ) or ( Mobile *entity* ) .

It is one of the key features of the QSM that it generally contains not only one interpretation of the scene, but a (possibly empty) *set* of interpretations which are all pursued simultaneously. At any point in time, a hypothesis is said to be "feasible" if it exists in the QSM and is not in conflict with some observation made since it was established.

Interpretations are structured as an inheritance network of partial hypotheses. Individual scene interpretations are treated as "closed worlds", i.e., a new conclusion only holds within an interpretation where all the required premises are true. Interpretations are also checked for internal consistency, e.g., entities cannot be both stationary *and* mobile within the same interpretation.

The QSM is maintained through a generate-and-test process as the core of a rule-based blackboard system. The two major groups of rules are: *Generation Rules* and *Verification Rules*.

*Generation Rules*

Generation rules examine the (derotated) image sequence for significant changes and modify each interpretation in the QSM. Some of these observations have unconditional effects upon the model. For example, if an image feature is found to be moving *towards* the Fuzzy FOE (instead of diverging away from it), then it belongs to a moving entity in 3D space. The actual rule contains only one premise and asserts (MOBILE ?x) as a global fact (i.e., it is true in every interpretation):

```
(defrule DEFINITE_MOTION
  (MOVING_TOWARDS_FOE ?x ?t)
  =>
  (at ROOT (assert (MOBILE ?x))))  /*a global fact*/
```

The directive "at ROOT" places the new fact at the root of the interpretation graph, i.e., it is inherited by all existing interpretations.

Other observations depend upon the facts that are currently true in a "world" and, therefore, may have only local consequences inside particular interpretations. For example, if two image features *A* and *B* lie on opposite sides of the Fuzzy FOE and they are getting closer to each other, then they must be in relative motion in 3D space. If an interpretation exists that considers at least one of the two entities (x,y) stationary, then (at least) the other entity cannot

be stationary (i.e., it must be mobile). The following rule "fires within" each interpretation that considers the first entity (x) stationary:

```
(defrule RELATIVE_MOTION
  (OPPOSITE_FOE ?x ?y ?t) /* first observation */
  (CONVERGING ?x ?y ?t)   /* second observation */
  (STATIONARY ?x) /* true inside an interpretation */
  =>
  (assert (MOBILE ?y)))   /* local to this interpretation */
```

While some image observations allow direct conclusions about motion in the scene, other observations hold cues about the stationary 3D structure. If the *exact* location of the FOE is known, then the depth of each stationary point (i.e., its 3D distance from the camera) is proportional to the rate of expansion (from the FOE) of its image (Equation 7). Consequently, for the Fuzzy FOE, where a set of potential FOE locations is given, the distance Z(*A*) of a stationary point *A* is determined as an interval instead of one single number:

$$\mathbf{Z}^{min}(A) \leq \mathbf{Z}(A) \leq \mathbf{Z}^{max}(A) .$$

Therefore, a point *A* is closer in 3D than another point *B*, if the corresponding ranges of depth do not overlap, i.e.,

$$\mathbf{Z}^{max}(A) < \mathbf{Z}^{min}(B) \rightarrow (CLOSER\ A\ B) .$$

Since this conclusion only holds if both features are actually stationary, the following rule fires only within a suitable interpretation (if it exists):

```
(defrule CLOSER_FROM_EXPANSION
  (STATIONARY ?x)          /*interpretation where */
  (STATIONARY ?y)          /*both are stationary */
  (< (Zmax ?x) (Zmin ?y))  /*no overlap in range */
  =>
  (assert (CLOSER ?x ?y))).
```

To compare the ranges of 3D points, another criterion can be used which does not require the rate of expansion from the FOE. Instead, the change of distances *between* certain pairs of features is observed. If two stationary points lie on the same side of the FOE and the distance between them is becoming smaller, then the *inner* feature (i.e., the one which is nearer to the FOE) is also closer in 3D space. This is a valuable test for features that are relatively near to each other in the image. It can be employed even if the image is not derotated and the location of the FOE is either only known very roughly or is completely outside the field of view (i.e., for a side-looking camera):

```
(defrule CLOSER_FROM_CHANGING_DISTANCE
  (STATIONARY ?x)          /*interpretation where */
  (STATIONARY ?y)          /*both are stationary, */
  (SAME_SIDE_OF_FOE ?x ?y) /*both on the right, */
  (CONVERGING ?x ?y)       /*dist. is shrinking */
  (INSIDE ?x ?y)           /*x is nearer to FOE */
  =>
  (CLOSER ?x ?y).
```

*Verification Rules*

While the purpose of the generation rules is to establish new hypotheses and conclusions, the purpose of *verification rules* is to review interpretations after they have been created and, if possible, prove that they are false. When a hypothesis is found to be inconsistent with some new observation, it is usually removed from the QSM. Any interpretation that is based on such a hypothesis is removed simultaneously. Since we are always trying to come up with a single (and hopefully correct) scene interpretation, this mechanism is important for pruning the search tree.

Verification rules are typically based on image observations that, used as generators, would produce a large number of unnecessary conclusions. For example, the general layout of the scene seen from the top of a land-based vehicle suggest the rule of thumb that things which are *lower* in the image are generally closer to the camera. Although this rule is not strong enough to draw direct conclusions, it may be used to verify existing hypotheses:

```
(defrule LOWER_IS_CLOSER_HEURISTIC
  (CLOSER ?x ?y) (BELOW_THE_HORIZON ?x ?t)
  (BELOW_THE_HORIZON ?y ?t) (BELOW ?y ?x ?t)
  =>
  /*mark this interpretation as conflicting*/
  (assert (CONFLICT LOWER/CLOSER ?x ?y))).
```

Whenever an existing hypothesis (CLOSER ?x ?y) violates the above rule of thumb, this rule fires and marks the interpretation as conflicting. How the conflict is eventually resolved depends upon the global state of the QSM. Simply removing the afflicted interpretation would create an empty model if this interpretation was the only one. This task is handled by a set of dedicated *conflict resolution rules*.[1]

The kind of rules described up to this point are mainly based upon the geometry of the imaging process, i.e., perspective projection. Other important visual clues are available from occlusion analysis, perceptual grouping, and semantic interpretation. *Occlusion* becomes an interesting phenomenon when features of higher dimensionality than points are employed, such as lines and regions. Similarities in form and motion found by *perceptual grouping* allow us to assemble simple features into complex objects. Finally, as an outcome of the recognition process, *semantic* information may help to disambiguate the scene interpretation. If an object has been recognized as a building, for example, it makes every interpretation obsolete that considers this object mobile. For all these various lines of reasoning, the QSM serves as a common platform.

*Meta Rules*

In summary, the construction of the QSM and the search for the most plausible scene interpretation are guided by the following meta rules:

- Always tend towards the "most stationary" (i.e. most conservative) solution. By default all new entities are considered stationary.
- Assume that an interpretation is feasible unless it can be proved to be false ( the principle of "lack of conflict").
- If a new conclusion causes a conflict in one but not in another current interpretation, then remove the conflicting interpretation.
- If a new conclusion cannot be accommodated by any current interpretation, then create a new, feasible interpretation and remove the conflicting ones.

## 5. EXPERIMENTAL RESULTS USING QSM

### 5.1 Fuzzy FOE Results

In the following, the results of the FOE-algorithm and computation of the vehicle's velocity over ground are shown on a real image sequence taken from the moving ALV. The original sequence was provided on standard video tape with a

frame-rate of 30 per second. Out of this original sequence, images were taken in 0.5 second intervals, i.e., at a frame rate of 2 per second in order to reduce the amount of storage and computation. The images were digitized to a spatial resolution of 512×512, using only the $Y$-component (luminance) of the original color signal.

Figure 34 shows the edge images of 16 frames with the points being tracked labeled with ascending numbers. We have developed an adaptive windowing technique as an extension of relaxation labeling disparity analysis for the selection and matching of tracked points.[9] The actual image location of each point is the lower left corner of the corresponding mark. The resulting data structure consisted of a list of point observations for each image (time), e.g.,

time $t_0$: ( $(p_1\ t_0\ x_1\ y_1)$ $(p_2\ t_0\ x_2\ y_2)$ $(p_3\ t_0\ x_3\ y_3)$ ... )

time $t_1$: ( $(p_1\ t_1\ x_1\ y_1)$ $(p_2\ t_1\ x_2\ y_2)$ $(p_3\ t_1\ x_3\ y_3)$ ... )

. . .

Points are given a unique label when they are encountered for the first time. After the tracking of a point has started, its label remains unchanged until this point is no longer tracked. When no correspondence is found in the subsequent frame for a point being tracked, either because of occlusion, or the feature left the field of view, or because it could not be identified, tracking of this point is discontinued. Should the same point reappear again, it is treated as a new item and given a new label. Approximately 25 points per image have been selected in the sequence shown in Figure 34.

In the search for the Focus of Expansion, the optimal FOE-location from the previous pair of frames is taken as the initial guess. For the very first pair of frames (when no previous result is available), the location of the FOE is guessed from the known camera setup relative to the vehicle. The points which are tracked on the two cars (24 and 33) are assumed to be known as moving and are not used as reference points to compute the FOE, vehicle rotation, and velocity. This information is eventually supplied by the reasoning processes in conjunction with the *Qualitative Scene Model*.

Figure 35 shows the results of computing the vehicle's motion for the same sequence as in the previous figure. Each frame $t$ displays the motion estimates for the period between $t$ and the previous frame $t-1$. Therefore, no estimate is available at the first frame (182). Starting from the given initial guess, the FOE-algorithm first searches for the image location, which is not prohibited and where the error function (equation 35) has a minimum.

The optimal horizontal and vertical shift resulting at this FOE-location is used to estimate the vehicle's rotations around the X- and Y-axis. This point, which is the initial guess for the subsequent frame, is marked as a small circle inside the shaded area. The equivalent rotation components are shown graphically on a $\pm$ 1° scale. They are relatively small throughout the sequence such that it was never necessary to apply intermediate derotation and iteration of the FOE-search. Along with the original displacement vectors (solid lines), the vectors obtained after derotation are shown with dashed lines.

After the location with minimum error has been found, it is used as the seed for growing a region of potential FOE-locations. The growth of the region is limited by two restrictions:

- The ratio of maximum to minimum error inside the region is limited, i.e., $E_n^i/E_n^{min} = \rho^i \leq \rho^{lim}$ (see equation 40 for the definition of the error function $E_n$). No FOE-location for which the error ratio $\rho^i$ exceeds the limit $\rho^{lim}$ is joined to the region. Thus the final size of the region depends on the shape of the error function. In this example, the ratio $\rho^{lim}$ was set at 4.0. Similarly, no *prohibited* locations (Figure 25) are considered.
- The maximum size of the region $M$ is given. he given FOE-region. region regardless of their error values. The resulting error ratio $\rho^{max} = \max(\rho^i)$ for the points inside the region indicates the shape of the error function for this area. A low value for the ratio $\rho^{max}$ indicates a flat error function. The value for $\rho^{max}$ is shown as *FOE-RATIO* in every image.

For the computation of absolute vehicle velocity, only a few prominent displacement vectors were selected in each frame pair. The criterion was that the vectors are located below the FOE and their length is more than 20 pixels. The endpoints of the selected (derotated) vectors are marked with dark dots. The parameter used for the computation of absolute advancement is the height of the camera above the ground, which is 3.3 meters (11 feet).

### 5.2 Motion Detection and Tracking

Following the computation of the FOE locations in each of the frames in the sequence, the QSM processes the images and determines the motion of the moving objects and builds a 3D representation of the environment as described in section 4. Figures 36 (a-f) show the complete scene interpretations starting at frame 183 up to frame 197. Interpretations are ranked by their number of stationary entities, i.e., "Interpretation 1" is ranked higher than "Interpretation 2" if both exist. During this run, the maximum number of concurrent interpretations was two. Whenever two interpretations exist at the same time, they are lined-up horizontally in Figure 36. Otherwise, interpretations are displaced to indicate that they refer to different points in time. Entities are marked as stationary or mobile. Entities which carry no mark (just the label) are stationary and have not been found to be closer than any other entity in the scene. A square without a pointer in any direction means that this entity is considered mobile, but that the direction of movement could not be determined for the current frame interval.

The scene contains two moving objects, a car (24) which has passed the ALV and is moving away throughout the sequence and another vehicle (33), approaching the ALV on the same road, which appears in frame 185.

After the first pair of frames (frame 183), two interpretations are created due to the movement of point 24 (the receding car). Interpretation 1 is preferred because it contains 23 stationary entities instead of 18 in interpretation 2. The latter interpretation is discarded due to inconsistent expansion of the points considered moving downwards.

A single interpretation is pursued from frame 184 until frame 194. In this period, no object motion other than the one caused by point 24 is observed. However, the perception of the 3D structure of the stationary part of the scene is continuously refined by adding new *closer*-relationships between entities. Point 24 is always considered mobile, although the direction of its movement cannot be identified between every pair of frames.

After frame 195, two interpretations again become feasible, this time caused by the movement of the approaching car (point 33). Again, the (correct) alternative 1 was ranked higher due to the larger number of stationary entities.

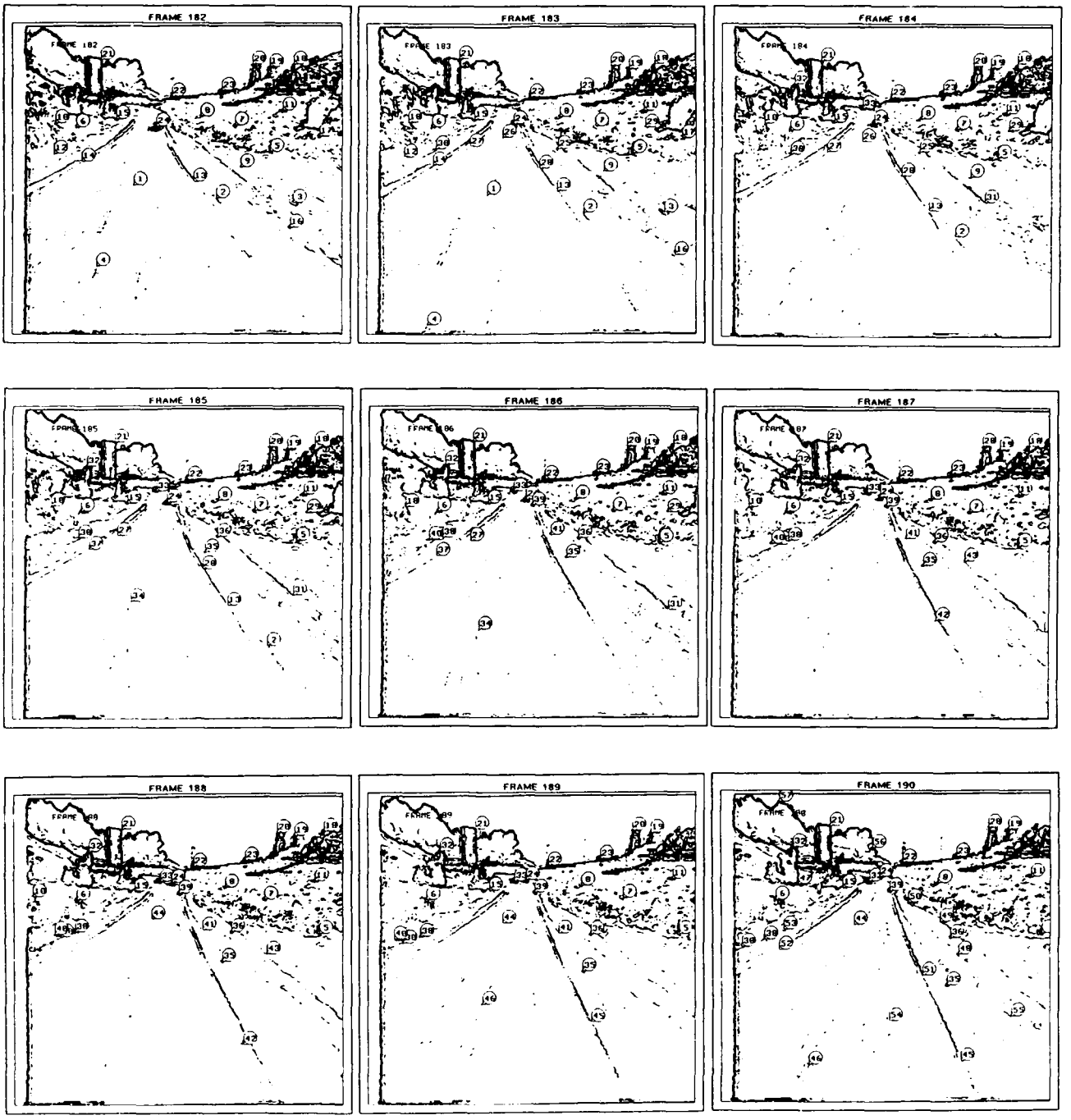



Figure 34: Original image sequence taken from the moving ALV after edge detection and point selection. The selected points are located at the lower-left corners of their marks. *(a)* Frames 182-190 of the original image sequence. The scene contains two moving objects, one car moving away from the ALV (point 24) and another car approaching the ALV (point 33).

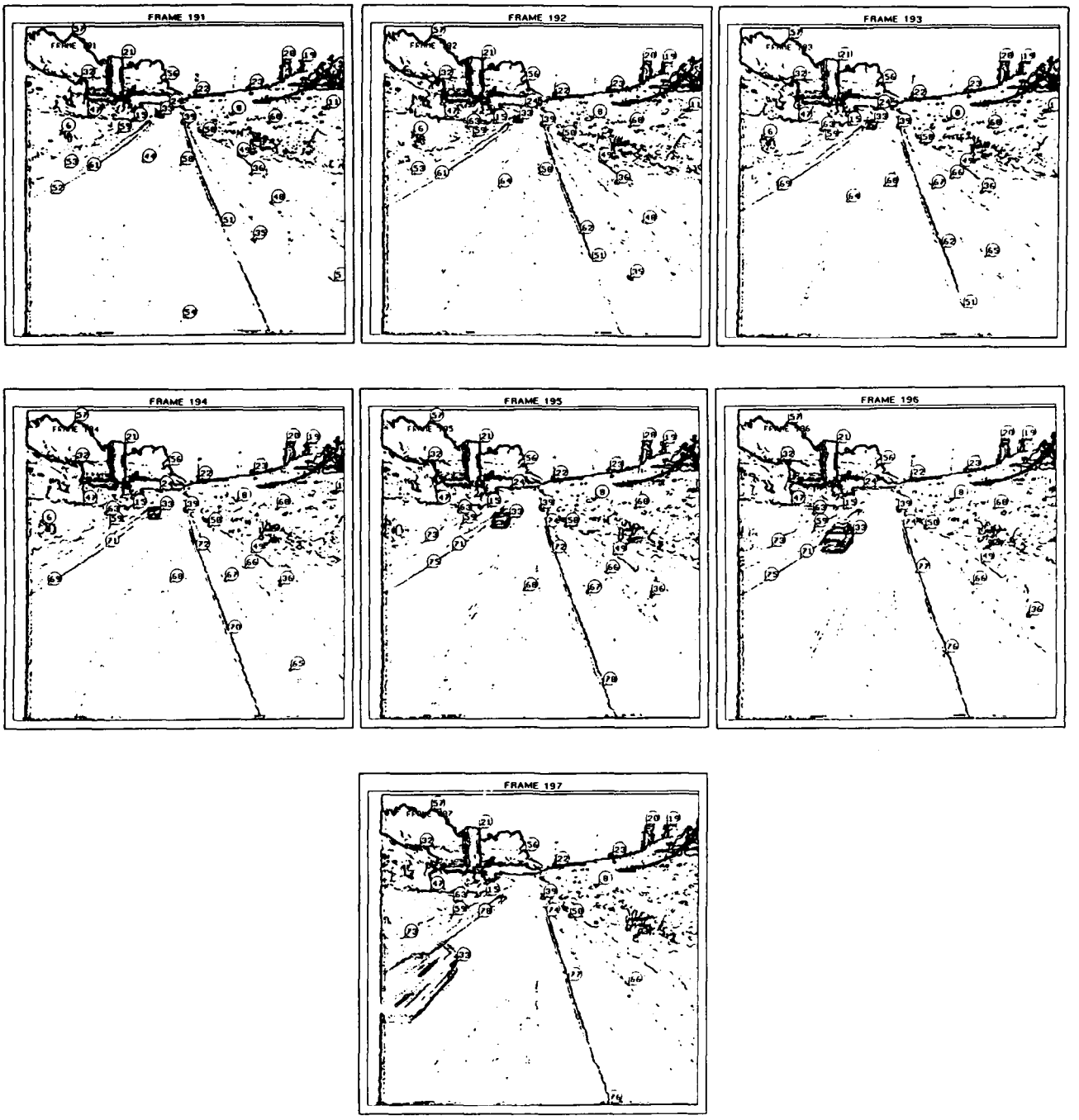



Figure 34(b): Frames 191-197 of the original image sequence after edge detection and point selection.

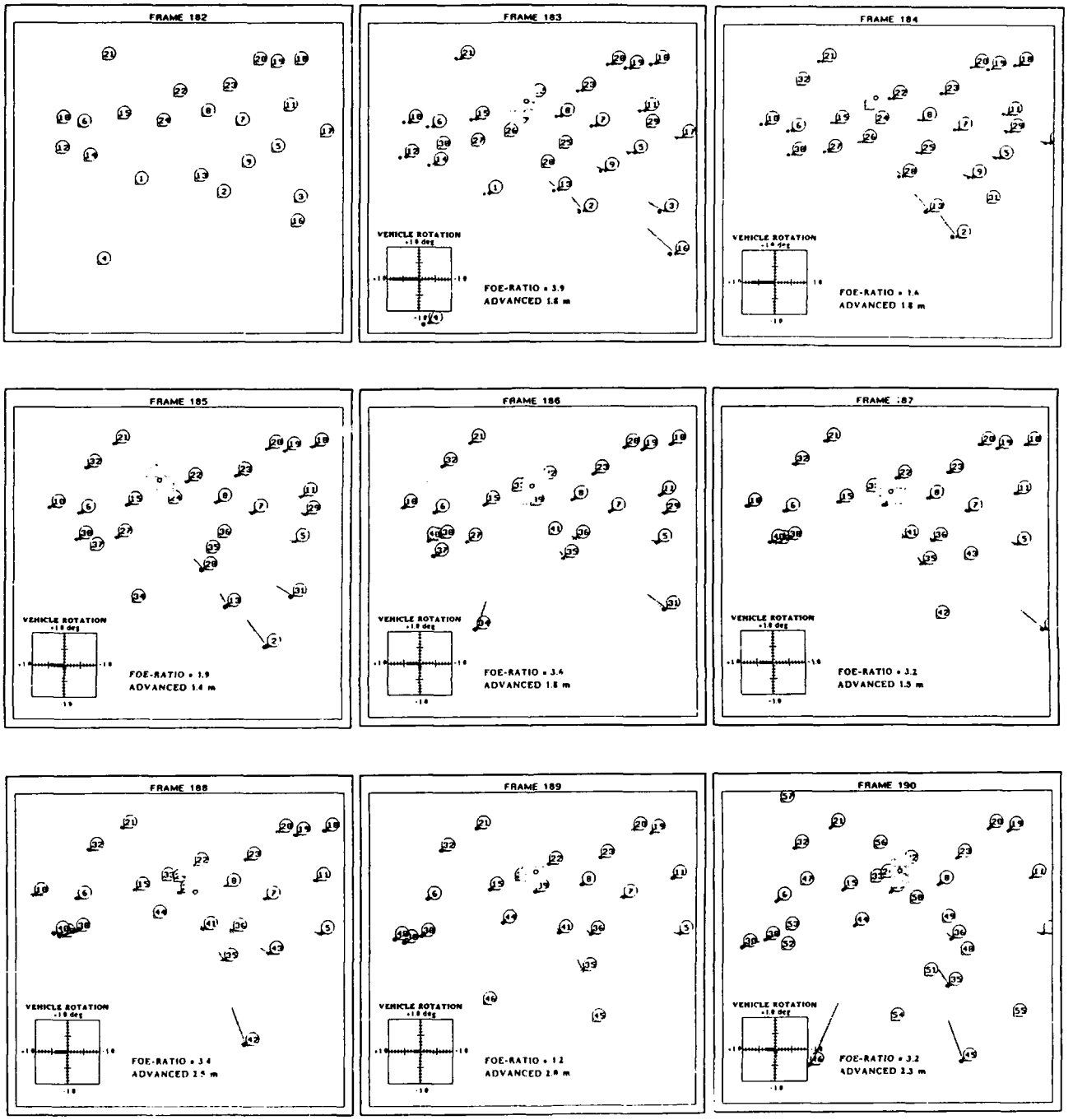



Figure 35: Displacement vectors and estimates of vehicle motion for the image sequence shown in Figure 34. The shaded area marks the possible FOE locations, the circle inside is the FOE with the lowest error value. *FOE-RATIO* measures the flatness of the error function inside this area. The absolute advancement of the vehicle is estimated in meters, vehicle rotation is plotted in a coordinate grid over ±1.0°. *(a)* Displacement vectors and estimates of vehicle motion for frames 182-190 shown in Figure 34(a).

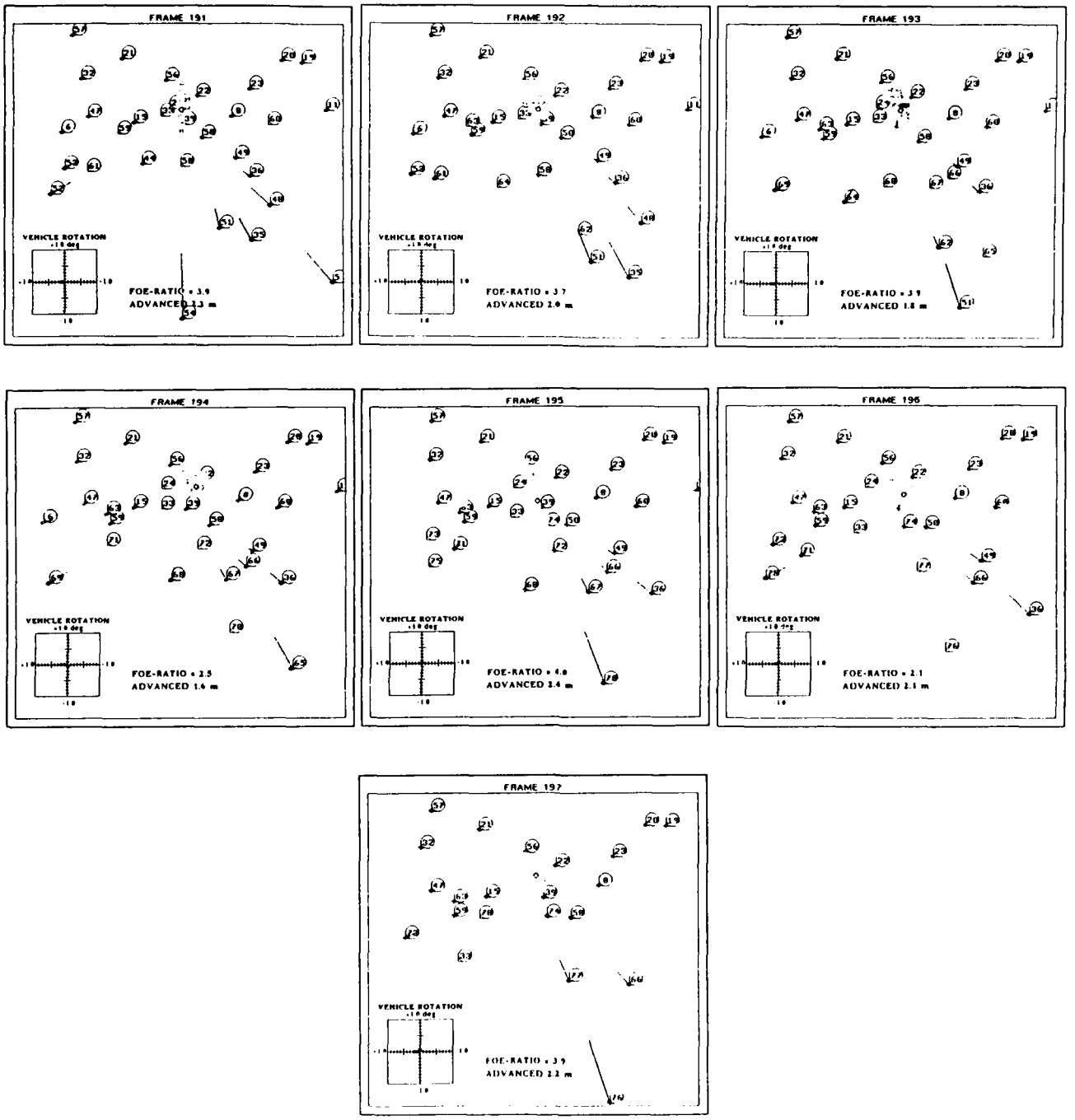



Figure 35(b): Displacement vectors and estimates of vehicle motion for frames 191-197 shown in Figure 34(b).

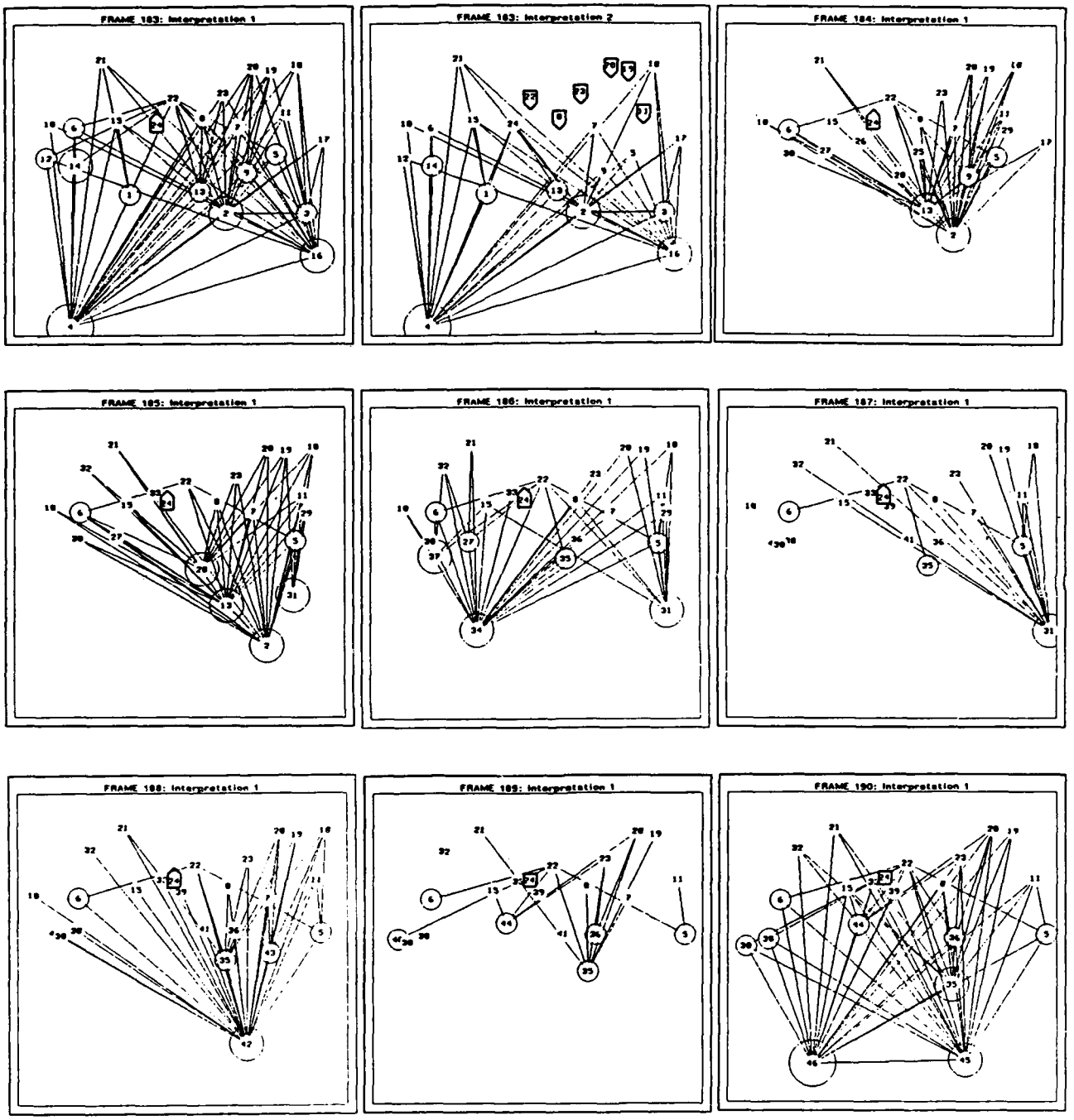



Figure 36: Scene interpretations for image sequence in Figure 34. *(a)* Frames 183-190. After the first pair of frames two interpretations are created due to the movement of point 24 (the receding car). Interpretation 1 is ranked higher because it contains 23 stationary entities instead of 18 in interpretation 2. The latter interpretation is discarded after frame 184 due to inconsistent expansion of the points considered moving downwards. The single interpretation from frame 184 is pursued, because no object motion other than the one caused by point 24 is observed in this period.

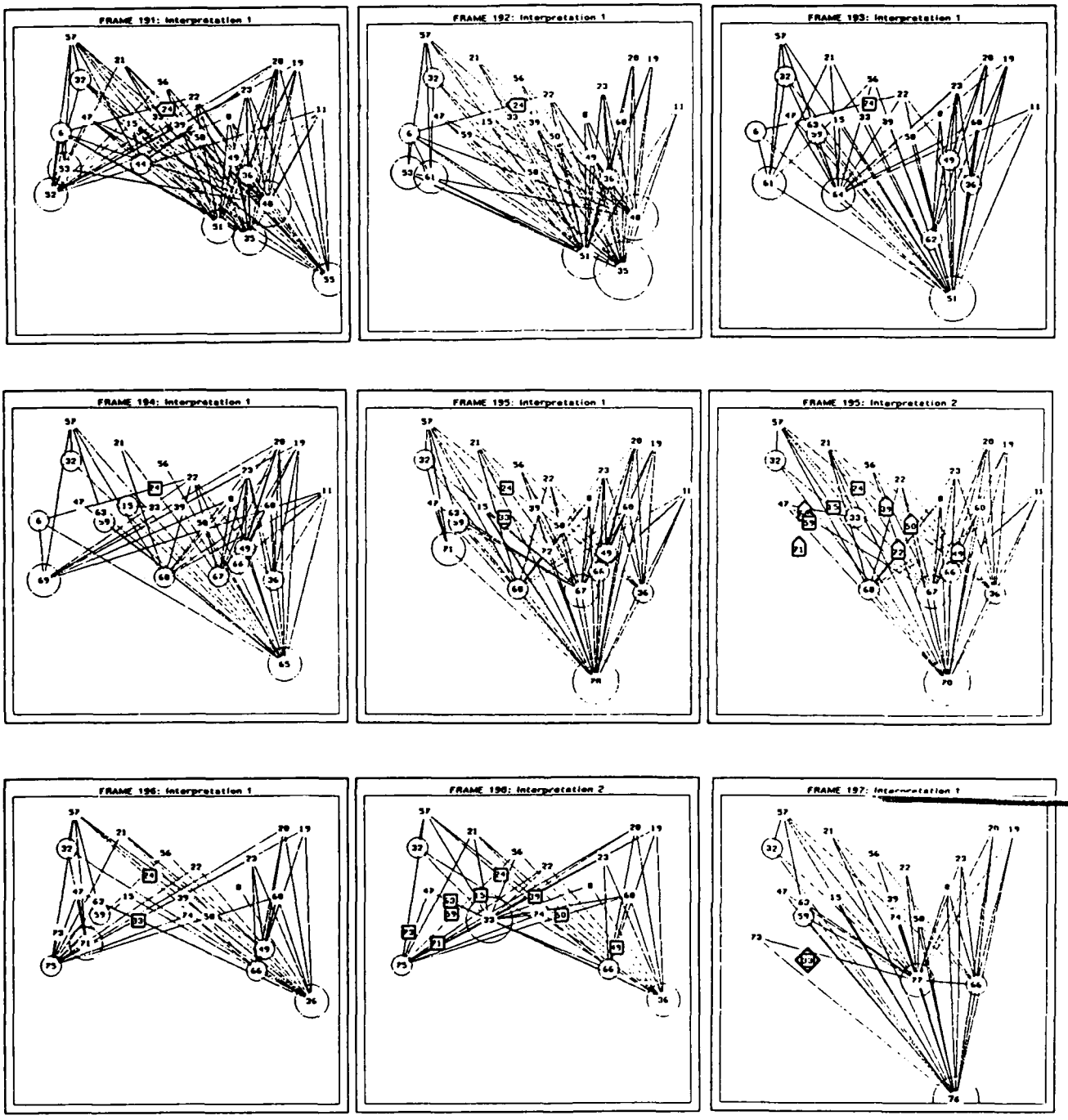



Figure 36: *(b)* Frames 191-197. The 3D structure of the stationary part of the scene is continuously refined by adding new *closer*-relationships between entities.

The two interpretations are pursued simultaneously until frame 197. At this time, a pending *closer*-conflict between point 33 and 76 in interpretation 2 is resolved. Point 76 is clearly closer to the ALV because it is near the bottom of the image, but the faster expansion of point 33 contradicts that. Therefore, as in the synthetic example, interpretation 2 can be discarded in favor of interpretation 1 and point 33 is correctly identified as approaching the ALV (Figure 36(f)).

This experiment shows that the *Qualitative Scene Model* is robustly maintained under real-world conditions, i.e., noise, distortion, imperfect derotation, and location of the FOE. The number of simultaneous interpretations is quite small (maximum is 2) and the correct interpretation clearly ranks higher at any point in time.

## 6. CONCLUSIONS

In this paper, we presented a qualitative approach to scene understanding for mobile robots in dynamic environments. The challenge of understanding such image sequences is that stationary objects do not appear as stationary in the image and mobile objects do not necessarily appear to be in motion. Consequently, the detection of 3D motion often requires reasoning far beyond simple 2D change analysis.

The approach taken here clearly departs from related work by following a strategy of qualitative, rather than quantitative, reasoning and modeling. All the numerical efforts are packed into the computation of the Focus of Expansion (FOE), which is is accomplished entirely in 2D. To cope with the problems of noise and errors in the displacement field, we determine a region of possible FOE-locations instead of a single FOE. Termed the *Fuzzy FOE*, it is probably one of the most robust techniques of this kind available today.

We showed on sequence of data that, even without knowing the exact location of the FOE, powerful conclusions about motion and 3D scene structure are possible. From these clues, we construct and maintain an internal 3D representation, termed the *Qualitative Scene Model*, in a generate-and-test cycle over extended image sequences. This model also serves as a platform for other visual processes, such as occlusion analysis, perceptual grouping, and object recognition. To overcome the ambiguities inherent to dynamic scene analysis, multiple interpretations of the scene are pursued simultaneously.

The examples given in the paper show the fundamental operation of our approach on real images produced by the Autonomous Land Vehicle. We also wanted to demonstrate that some apparently simple situations require relatively complex paths of reasoning. Of course, the exclusive use of displacement vectors from point features is a limiting factor. To exploit a larger part of the information contained in the image and to demonstrate the full potential of our approach, more complex 2D features, such as lines and regions, will be employed in our future work.

## REFERENCES


1. B. Bhanu and W. Burger, "DRIVE: Dynamic Reasoning from Integrated Visual Evidence," *Proc. DARPA Image Understanding Workshop*, pp. 581-588 Morgan Kaufmann Publishers, (Feb. 1987).
2. B. Bhanu and W. Burger, "Approximation of Displacement Field Using Wavefront Region Growing," *Computer Vision, Graphics and Image Processing*, (March 1988).
3. S. Bharwani, E. Riseman, and A. Hanson, "Refinement Of Environmental Depth Maps Over Multiple Frames," Proc. IEEE Workshop on Motion, Kiawah Island Resort, pp. 73-80 (May 1986).
4. R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York (1973).
5. O.D. Faugeras, F. Lustman, and G. Toscani, "Motion and Structure from Point and Line Matches," Proc. of 1st International Conference on Computer Vision, pp. 25-34, London (June 1987).
6. R.M. Haralick, "Using Perspective Transformations in Scene Analysis," *Computer Graphics and Image Processing* **13** pp. 191-221 (1980).
7. E.C. Hildreth and N.M. Grzywacz, "The Incremental Recovery of Structure from Motion: Position vs. Velocity Based Formulations," Proc. IEEE Workshop on Motion, Kiawah Island Resort, pp. 137-143 (May 1986).
8. R. Jain, "Direct Computation of the Focus of Expansion," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-5**(1) pp. 58-64 (January 1983).
9. J. Kim and B. Bhanu, "Motion Disparity Analysis Using Adaptive Windows," Technical Report 87SRC38, Honeywell Systems & Research Center (June 1987).
10. H.P. Moravec, "Towards Automatic Visual Obstacle Avoidance," Proc. 5th International Joint Conference on Artificial Intelligence, pp. 584 (August 1977).
11. H.-H. Nagel, "Image Sequences - Ten (octal) Years - From Phenomenology towards a Theoretical Foundation," Proc. Intern. Conf. on Pattern Recognition, Paris, pp. 1174-1185 (1986).
12. K. Prazdny, "Determining the Instantaneous Direction of Motion from Optical Flow Generated by a Curvilinear Moving Observer," *Computer Graphics and Image Processing* **17** pp. 238-248 (1981).
13. K. Prazdny, "On the Information in Optical Flows," *Computer Vision, Graphics, and Image Processing* **22** pp. 239-259 (1983).
14. D. Regan, K. Beverly, and M. Cynader, "The Visual Perception of Motion in Depth," *Scientific American*, pp. 136-151 (July 1979).
15. J.H. Rieger, "Information in Optical Flows Induced by Curved Paths of Observation," *J. Opt. Soc. Am.* **73**(3) pp. 339-344 (March 1983).
16. W.B. Thompson and J.K. Kearney, "Inexact Vision," Proc. IEEE Workshop on Motion, Kiawah Island Resort, pp. 15-21 (1986).
17. R.Y. Tsai and T.S. Huang, "Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects with Curved Surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-6**(1) pp. 13-27 (January 1984).
18. S. Ullman, *The Interpretation of Visual Motion*, MIT Press, Cambridge, Mass. (1979).
19. A. Verri and T. Poggio, "Qualitative Information in the Optical Flow," Proc. DARPA Image Understanding Workshop, Los Angeles, pp. 825-834 (February 1987).

# QUALITATIVE NAVIGATION II

Tod S. Levitt, Daryl T. Lawton, David M. Chelberg,
Kerry V. Koitzsch, and John W. Dye

Advanced Decision Systems
201 San Antonio Circle, Suite 286
Mountain View, CA 94040-1289

## ABSTRACT

We have developed a multi level theory of representation of large scale space based upon the observation and re-acquisition of distinctive visual events, i.e. landmarks. The theory provides a method for integrating the visual information from many images into a uniform database as a sensor based robot moves through its environment. It incorporates the outputs of tracking of significant perceptual events, as well as providing a high level visual description of places that can be used to robustly guide a robot with visual predictions even with of a poorly stabilized sensor platform. The representation provides the theoretical foundations for visual memory databases and path planning and execution algorithms that include coordinate free, topological representation of relative spatial location, yet smoothly integrate available metric knowledge of relative or absolute angles and distances. In order to demonstrate our claims we have built a qualitative navigation simulator, called the QUALNAV model, that provides a software laboratory for experimenting with spatial relationships in visual memory and their relationship to vision-based path planning and execution.

## INTRODUCTION

We have developed a multi level theory of representation of large scale space based upon the observation and re-acquisition of distinctive visual events, i.e. landmarks. The theory provides a method for integrating the visual information from many images into a uniform database as a sensor based robot moves through its environment. It incorporates the outputs of tracking of significant perceptual events, as well as providing a high level visual description of places that can be used to robustly guide a robot with visual predictions even with a poorly stabilized sensor platform. The representation provides the theoretical foundations for visual memory databases and path planning and execution algorithms that include coordinate free topological representation of relative spatial location, yet smoothly integrate available metric knowledge of relative or absolute angles and distances. Rules and algorithms have been developed that, under the assumption of *correct association of landmarks on re-acquisition (although* not assuming landmarks are necessarily re-acquired) provide a robot with navigation and guidance capability. The ability to deduce or update a map of large scale space, a posteriori, is a by-product of the inference process. In order to demonstrate our claims, we have built a qualitative navigation simulator, called the QUALNAV model, that provides a software laboratory for experimenting with spatial relationships in visual memory and their relationship to vision-based path planning and execution.

The heart of the QUALNAV model is a symbolic representation of *large scale space that is derived from* visual landmark observations. There is topological localization information present in the ordinal sequence of landmarks, there is a sense in which we can compute differences between geographic regions, and observe which region we are in. The basic concept is to note that if we draw a line between two (point) landmarks, and project that line onto the (possibly not flat) surface of the ground, then this line divides the earth into two distinct regions. If we can observe the landmarks, we can observe which side of this line we are on. The "virtual boundary" created by associating two observable landmarks together thus divides space over the region in which both landmarks are visible. We call these landmark pair boundaries (LPBs), and denote the LPB constructed from the landmarks $L_1$ and $L_2$ by LPB($L_1, L_2$). The observable relative orientation of the landmarks, i.e., the left to right order of the pair of landmarks, indicates which side of the landmark pair boundary (LPB) we are on. A set of LPBs with orientations determines a region on the ground called an orientation region (OR). Orientation regions are, in turn, derivable from sets of commonly visible landmarks at a place.

A place, as a point on the surface of the ground, is defined by the landmarks and spatial relationships between landmarks that can be observed from a fixed location. Data about places is stored in a data structure called viewframes.

Viewframes provide a definition of place in terms of relative angles and angular error between landmarks, and

very coarse estimates of the absolute range of the landmarks from the point of observation. Boundaries and orientation regions provide a more qualitative definition of place. Both concepts allow us to localize ourselves in space relative to a set of observed landmarks, without necessarily using a priori map data.

In using viewframes to localize our position in space, we make use of observed or inferred data about our approximate range to landmarks. Errors in ranging and relative angular separation between landmarks are smoothly accounted for. A priori map data can also be incorporated. In [LLCN87], we presented a closed form solution for all possible triangulations among a set of landmarks with range estimates, that serves as a mathematical foundation for local coordinate systems based on observations of those landmarks. This viewframe-level theory was brought up in the QUALNAV simulator model, and was empirically shown to be remarkably robust in the face of very poor range and angular measurements. Sensitivity curves were presented in [Lev87].

The planning done in QUALNAV to demonstrate the usefulness of the visual memory representation, depended on the range estimates made in viewframes. However, if we drop all range information, we can still use the notion of boundaries to determine our qualitative position relative to other landmarks. LPBs can be derived by considering pairs of landmarks in viewframes; in this case we speak of the set of orientation regions induced by or associated to the viewframe.

Basic approaches to reasoning about path planning in visual memory over orientation regions was presented in [LLCN87]. The QUALNAV simulator now includes orientation region planning and execution. Results are shown in this paper. Two additional theoretical developments are presented. Using translational motion processing results of Lawton [Law83] we have shown that the approximate distance until contact with an LPB can be robustly recovered. This, combined with some additional trigonometry based on observed angles, allows approximate map recovery from visual memory which in turn allows sequential prediction of LPB crossing. Also, if a robot carries a compass and makes purely local (non-cumulative) direction readings at viewframes, it is possible to do path planning over orientation regions in visual memory without making any range estimates. The algorithm we present is parallel, and linear time in the graph diameter of visual memory.

## ORIENTATION REGION REPRESENTATIONS

As a robot moves over the ground surface, the crossings of LPBs indicates passage from one orientation region to another. Thus, orientation regions are bounded by the unique observable visual events of passing to the right of, left of, or between a pair of landmarks. In this manner, LPBs and orientation regions yield a natural notion of headings and paths in the environment.

Figure 1(a) shows the LPBs that are implicit in a viewframe where landmarks $L_1$, $L_2$, $L_3$, $L_4$, and $L_5$ are simultaneously visible. The solid lines are the virtual boundaries created by landmark pairs. Figure 1(a) can be misleading in that it seems to imply that the LPBs contain the data to compute the angle-distance geometry of the sensor location relative to the landmarks. Figure 1(b) shows a representation of the same viewframe. Here the ranges to the landmarks have been changed, but the ordinal angular relationships between landmarks have not. The angle-distance geometry of our apparent location relative to the landmarks is completely different; however, the topological information, that is, the number of regions, their number of sides and adjacency relations, are preserved.

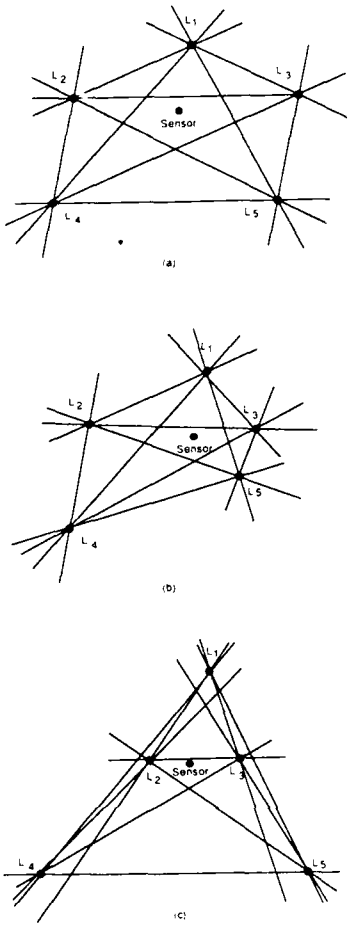



Figure 1: Viewframe Orientation Regions (a-c)

Roughly speaking, if we observe that landmark $L_1$ is on our left hand, and landmark $L_2$ is on our right, and the angle from $L_1$ to $L_2$ (left to right) is less than $\pi$ radians, then we denote this side of, or equivalently, this orientation of, the LPB by $[L_1 L_2]$. If we stand on the other side of the boundary, LPB($L_1, L_2$), "facing" the boundary, then $L_2$ will be on our left hand and $L_1$ on our right and the angle between them less than $\pi$ radians, and we can denote this orientation or side as $[L_2 L_1]$(left to right). The orientation

of the LPB relative to sensor position can be computed by:

$$= \text{sign}(\pi - \Theta_{12}) = \begin{cases} +1 \text{ if } \Theta_{12} < \pi \\ 0 \text{ if } \Theta_{12} = \pi \\ -1 \text{ if } \Theta_{12} > \pi \end{cases}$$

where $\Theta_{12}$ is the relative azimuth angle between $L_1$ and $L_2$ measured in an arbitrary sensor-centered coordinate system. Here, an orientation of +1 corresponds to the $[L_1 L_2]$ side of LPB $(L_1, L_2)$, -1 corresponds to the $[L_2 L_1]$ side of LPB$(L_1, L_2)$ and 0 corresponds to being on LPB$(L_1, L_2)$. It is straightforward to show that this definition of LPB orientation *does not depend on the choice of sensor-centered coordinate system.*

Figure 1(c) shows that the distance of the sensor to different LPBs is ambiguous. Note that, for example, the LPB $(L_1, L_4)$ is closer to the sensor in 1(c) than the LPB $(L_1, L_2)$, but is farther away in figure 1(b), even though the observed relative angles between landmarks is identical.

LPBs give rise to a topological division of the ground surface into observable regions of localization, called orientation regions. Crossing boundaries between orientation regions leads to a qualitative path planning based on perceptual information.

If a viewframe contains the landmark sequence $L_1$, $L_2$, $L_3$, $L_4$, then LPB$(L_1, L_3)$ and LPB$(L_2, L_4)$ must cross. It is possible for LPB$(L_1, L_2)$ and LPB$(L_3, L_4)$ to be parallel, and crossing points of greater than multiplicity 1 can happen; however, in natural environments these are relatively rare events (N.B. These assumptions do not hold in a city or many manmade environments). It follows that a reasonable estimate of the number of orientation regions implicit in a viewframe can be gotten by assuming that all LPB crossings generated by pairs of landmarks in the viewframe have multiplicity 1 (except for the crossings through landmarks themselves, which have multiplicity K-1 for K landmarks), and that no LPBs are parallel. Under these assumptions, the number of orientation regions determined by a viewframe only depends on the number of landmarks in the viewframe. The number of orientation regions implicit in a viewframe can be computed as:

number-of-orientation-regions

$$(K^4)/8 \quad (3K^3)/4 + (23K^2)/8 \quad (13K)/4 + 1$$

The conjunction of LPB boundaries observed from our current location defines a symbolic description of visually distinguishable geographic regions. Any conjunction of LPB orientations that shares at least one LPB, but has the orientation reversed, must be in a different region of space. The localization sensitivity of orientation regions for landmarks placed on a circle of diameter 100, is pictured in figure 2.

Now if we observe a viewframe and ignore the range information, we can still observe all $\binom{n}{2}$ LPBs created by taking landmarks in pairs. The conjunction of the orientations defines the boundaries of the orientation region we currently occupy. Any conjunction of LPB orientations that shares at least one LPB, but has the orientation reversed, must be in a different region of space.

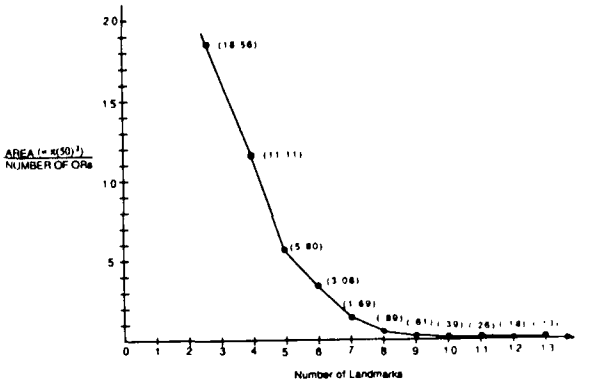



Figure 2: Orientation Region Localization Sensitivity

## QUALITATIVE PATH PLANNING OVER ORIENTATION REGIONS

As a landmark-recognizing vision system moves *through large scale space, it builds up a visual memory* consisting of interlocking sequences of orientation regions traversed through. Adjacency of orientation regions in visual memory can be determined by sharing a common, but opposite orientation, LPB. If two regions have a common boundary, it is possible to move between them by tracking the landmarks as we move toward the boundary. Thus, visual memory is an undirected graph where nodes are orientation regions and arcs join adjacent regions.

The following algorithm for qualitative path planning assumes that we carry a compass as we move through the environment and mark the direction North relative to observed landmarks, at each viewframe and as we cross each LPB. Note that this is a purely local sensor reading. It is not necessary to record changes in bearing as the sensor moves. Now we have at each viewframe (which is a node in the planning algorithm) the compass heading of each landmark. The start and goal points are orientation regions. We want to plan a path in our memory of large scale space between these regions.

We accomplish this by propagating paths outward from each of the start and goal regions. Each adjacent region to the start or goal region is an approximate compass heading away from the start or goal region respectively. We store this initial direction for each adjacent region. From each adjacent region, we now choose the single region adjacent to it that is a relative compass heading most close to the initial step from the start or goal. Thus, we propagate a single path for each region adjacent to the

start or goal region.

Because we minimize the compass difference from the original (not previous) step from start or goal as we propagate a path, there is a straight path propagating from each region adjacent to the start or goal node. Because there is more than one region adjacent to the start and goal modes, there are at least four paths, not all of which can be parallel. If a path hits a boundary node of visual memory, the path is repropagated to all regions adjacent to the boundary node. It follows that at least two of these paths, one from the start, one from the goal, must cross in visual memory. (This can be made a rigorous proof.) We check for crossing at each step by checking (but not propagating a path from) each adjacent region to see if a start or goal path has already crossed there.

This algorithm is illustrated in figure 3. Figure 3(a) shows the visual memory representation of large-scale space, with start and goal nodes, figure 3(b) shows the adjacent regions propagated to. Figure 3(c) and (d) shows the choice from each adjacent region to one that minimizes the heading difference from the initial step. Figure 3(e) shows propagation until path crossing is detected. Figure 3(f) shows the inferred path through adjacent orientation regions. This algorithm is clearly order linear in the graph-diameter of visual memory of large-scale space.

This path can be iteratively refined numerous ways, such as estimating the heading between start and goal and replanning using this knowledge. Due to space limitations, we do not explore these possibilities here.

A plan in visual memory is now a sequence of adjacent orientation regions. In path execution, we cross LPBs by tracking landmarks of the boundary LPBs between adjacent orientation regions. If we see landmarks further along our plan, we jump ahead to cross through them, thereby short-cutting the path following process opportunistically.

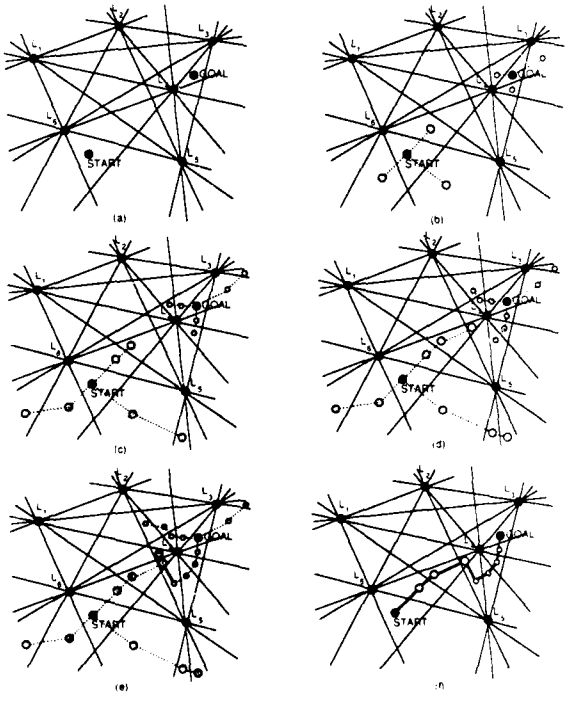



Figure 3: Qualitative Planning Algorithm (a-f)

## VISION-BASED PATH EXECUTION

Orientation-headings are conjunctions of specifiers for crossing LPBs. An orientation-destination-goal is a conjunction of LPB crossing specifiers, with no more than one crossing specifier per LPB in the conjunction. The termination condition corresponding to an orientation-heading is that all crossings have occurred. Termination can also occur if it is impossible to proceed without re-crossing an already crossed LPB, or if none of the LPBs can be located. A typical desired behavior for choosing an orientation heading is to steer for the angular bisector between the pair of landmarks that are output from the production system. It can be shown that the path is a hyperbola, whose foci are the pair of landmarks. This is pictured in figure 4. Under this scheme for robot guidance, the path of execution of a LPB plan is piecewise hyperbolic.

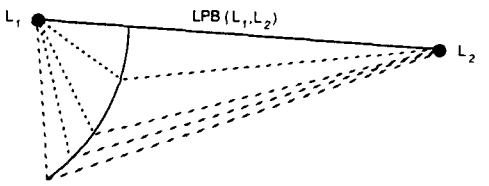
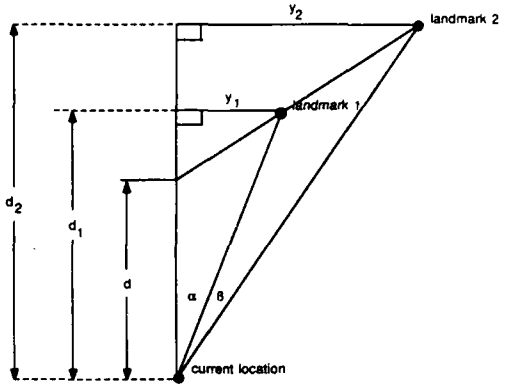



Figure 4: Hyperbolic LPB Crossing

The case when we are already on the goal side of an LPB requires more involved reasoning. We want to cross certain LPBs without crossing the ones we are already on the correct side of. Without (implicit or explicit) range information, we cannot tell which LPB we will cross first (on any heading). If we can estimate the distance until we are perpendicular to a tracked landmark relative to our direction of (linear) motion, then we make estimates for each landmark in an LPB, we can recover the approximate distance to an LPB crossing as shown in figure 5. Here, $\alpha$ and $\beta$ are observed angles. If d1, d2, y1, and y2 can be estimated, then the distance to the LPB, d, can be computed as shown in the figure. The following results of Lawton [Law83] are used to recover d1, d2, y1, and y2.

For pure translational motion, image-displacement paths are determined by the intersection of the translational axis with the image plane. If the translational axis intersects the image plane on the positive half of the axis, the point of intersection is called a *focus of expansion* (FOE) and the image motion is along straight lines radiating from it. This corresponds to sensor motion toward the FOC. This corresponds to camera motion away from observed landmarks. The intersections of axes parallel to




$$d = d_1 - y_2\cot(\alpha+\beta)+\sin^{-1}\left\{\frac{(d_1^2+y_1^2)^{\frac{1}{2}}\sin\beta}{(d_1^2+y_1^2+y_2^2+d_2^2-2(d_1^2+y_1^2)^{\frac{1}{2}}(y_2^2+d_2^2)^{\frac{1}{2}}\cos\beta)^{\frac{1}{2}}}\right\}$$

Figure 5: Distance to LPB Crossing

the image plane are points at infinity and thus may be considered to be either an FOE or an FOC in opposite directions.

Given the direction of translation and the image displacements of a set of environmental points, the relative depths of these points can be computed by solving the inverse perspective transform [RA76]. Relative depth can also be inferred from the position of a feature and the extent of its displacement relative to an FOE or an FOC. This relation is expressed as

$$\frac{d}{\Delta d} = \frac{i}{\Delta i}$$

where $d$ is the distance from the sensor focal plane of an environmental point at time $t+1$, $\Delta d$ is the extent of environmental displacement along the $d$ axis from time $t$ to time $t+1$, $i$ is the distance from the corresponding image point from the FOE or FOC at time $t$, and $\Delta i$ is the displacement of the image point from time $t$ to time $t+1$. Thus, the $d$ value of an environmental point can be recovered from image measurements in units of $\Delta d$, or what has been termed *time-until-contact* by Lee [Lee80].

## QUALNAV SIMULATOR RESULTS

The QUALNAV model has four logical environmental components, pictured in figure 6. The simulator level has a grid-based elevation map in which it keeps track of actual landmark locations and performs line of sight calculations to simulate the robot's visual system. The planning level occurs over the robot's visual memory. Memory of visual events are recorded with simulated range and angular error factored in for landmark sightings as the robot moves through the world. Actual robot motion and vision is simulated at the execution level. Headings at this level are computed relative to visual sightings. These headings

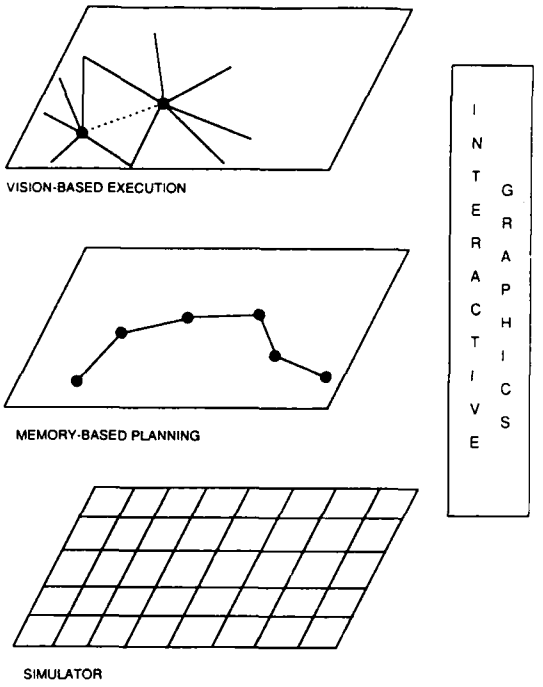
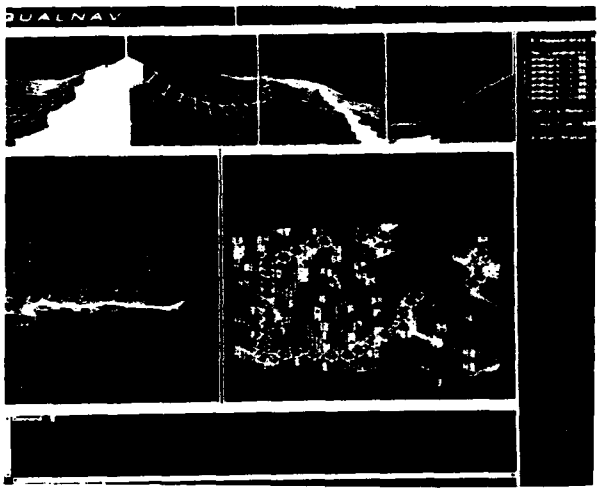



Figure 6: Logical QUALNAV Components

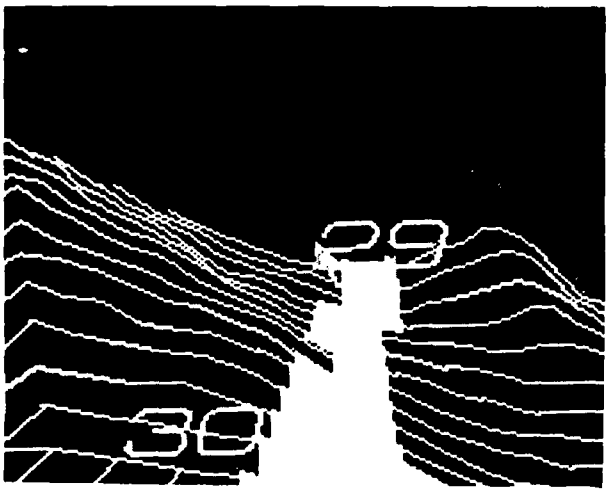
are translated to grid locations at the simulator level, and the robot move is executed. Then the new line of sight landmark sightings are returned to the planning level before control is returned to the execution level. Parameters such as landmark locations at the simulator level, maximum visual resolution at the planning level, and step-size for robot motion at the execution level, can all be interactively adjusted. For experiments, start and goal locations, location of landmarks, choice of displays and other control options, are all graphically selectable.

The QUALNAV displays are pictured in figure 7. Figure 7(a) shows the full display capability. The right most window allows menu selection of processing and also displays incremental vision, planning and motion results. The four topmost windows show the 360° robot's eye perspective view of the world. Figure 7(b) shows this view to the north only from a point in visual memory. The world consists of an elevation grid with landmarks, indicated by numbers, placed on it. The lower left window, and figure 7(c) show an overhead world view including, elevation contour lines, landmarks, and paths from memory, planning and execution. The lower right window, and figure 7(d) is a shaded orthographic view of the world, with robot vehicle icons showing the points in the world that the robot has passed through and recorded views of in visual memory.

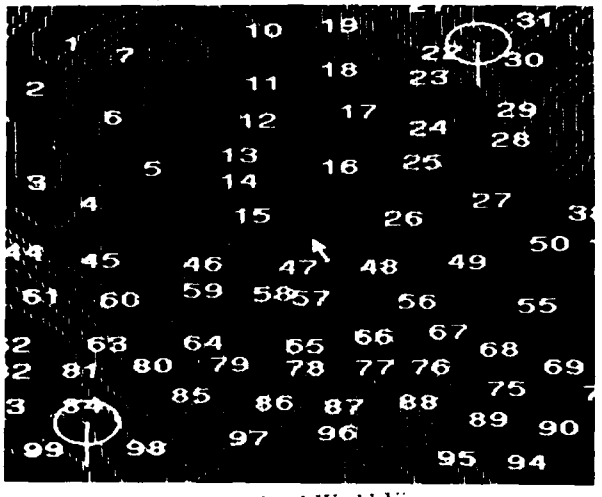
Figure 8 shows a typical run of the QUALNAV simulator. Figure 8(a) shows the start, and goal locations. Figure 8(b) shows the path already recorded in visual memory. Note that because of the path taken earlier,



(a) Full Display

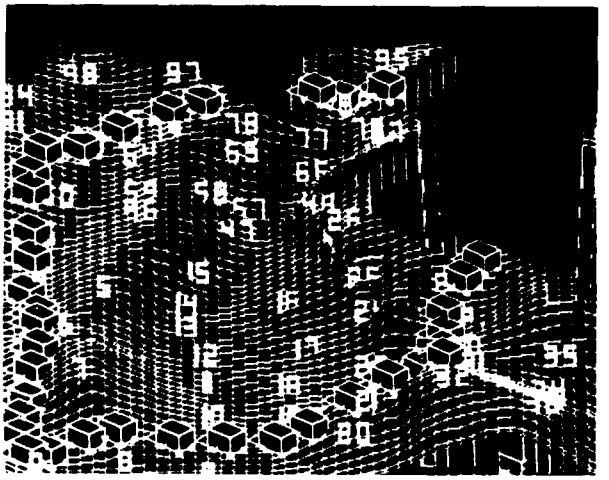


(b) Robot's-Eye Perspective View



(c) Overhead World View

Figure 7: QUALNAV Environment (a-c)



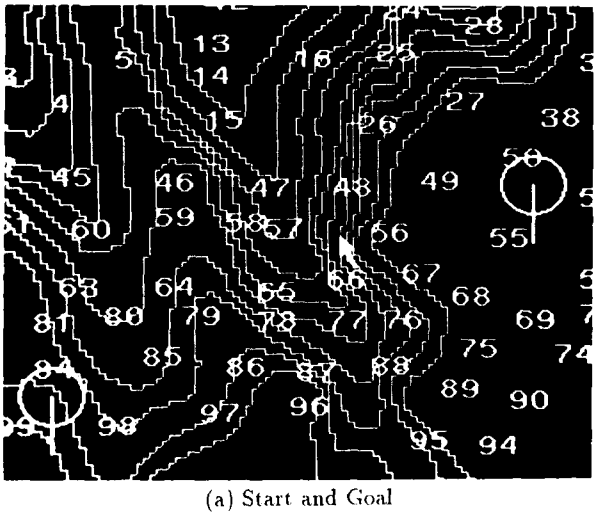
(d) Shaded z-buffer World View

Figure 7: QUALNAV Environment (d)(*cont.*)

the robot does not plan a more direct route to its goal, but instead, computes the viewframe plan of figure 8(c).

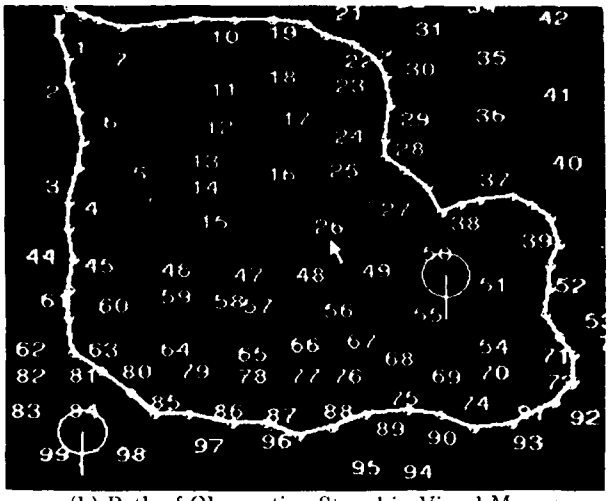
Figure 8(d) shows both the LPBs planned and observed and crossed in the course of executing the plan of figure 8(c). Notice that because landmarks 54 and 55 are not visible from the same viewframe as the goal location (near landmark 50), the plan did not cross LPB (54,55). However, in plan execution, the robot opportunistically observed that it would have to be on the [54,55] side of LPB (54,55) to be at its goal location, because the last viewframe in the plan was on that side of the LPB.

After the robot crosses LPB (54,55), it happens that its goal location is visible, so it heads directly toward it. Figure 8(e) shows the vision-based short-cut between the planned and executed paths.

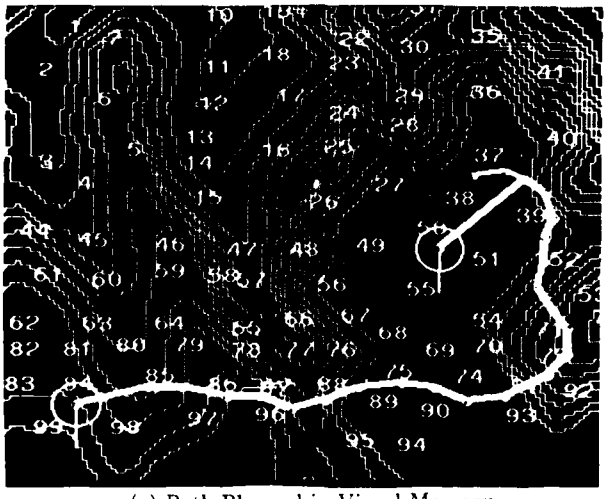


(a) Start and Goal

Figure 8: QUALNAV Results (a)



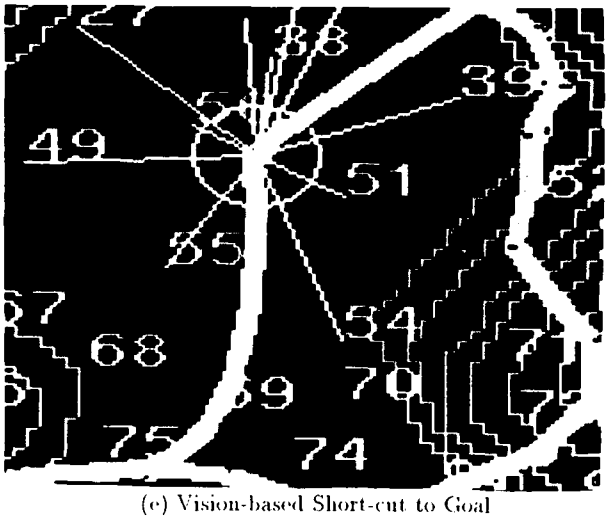
(b) Path of Observation Stored in Visual Memory



(c) Path Planned in Visual Memory

(d) LPBs Planned Versus LPBs Crossed During Execution

Figure 8: QUALNAV Results (b-d)(cont.)



(e) Vision-based Short-cut to Goal

Figure 8: QUALNAV Results (e)(cont.)

## REFERENCES


[BKM86] R. Bajcsy, E. Krotkov, and M. Mintz. *Models of Errors and Mistakes in Machine Perception.* Computer and Information Science Technical Report MS-CIS-86-26, GRASP LAB 64, University of Pennsylvania, 1986.

[EW85] R. Eastman and A. Waxman. Disparity functionals and stereo vision. In *Proceedings Image Understanding Workshop*, pages 245–254, Miami Beach, Florida, December 1985.

[FS87] N. Foreman and R. Stevens. Relationships between the superior colliculus and hippocampus: neural and behavioral considerations. *Behavioral and Brain Sciences*, 10(1):101–151, March 1987.

[HK85] M. Hebert and T. Kanade. First results on outdoor scene analysis using range data. In *Proceedings Image Understanding Workshop*, pages 224–231, Miami Beach, Florida, December 1985.

[KB77] L. T. Kozlowski and K. J. Bryant. Sense of direction, spatial orientation and cognitive maps. *Journal of Experimental Psychology: Human Perception and Performance*, 3:590–598, 1977.

[Kui78] B. J. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129–153, 1978.

[Kui82] B. J. Kuipers. The "map in the head" metaphor. *Environment and Behavior*, 14(2):202–220, March 1982.

[Kui85] B. J. Kuipers. *The Map-Learning Critter.* Technical Report AITR85-17, Austin, Texas, December 1985.

[Law83] D. T. Lawton. Processing translational motion sequences. *Computer Vision, Graphics, and Image Processing*, 22:116–144, 1983.

[Lee80] D. N. Lee. The optical flow field: the foundation of vision. *Philosophical Transactions of the Royal Society*, (B 208):385–397, 1980.

[Lev87] T. Levitt. Visual memory structure for a mobile robot. In *Proceedings of the IEEE Workshop on Spatial Reasoning and Multisensor Fusion*, Morgan Kaufmann Publishers, Los Altos, CA, October 1987.

[LK84] B. Lucas and T. Kanade. Optical navigation by the method of differences. In *Proceedings Image Understanding Workshop*, pages 272-281, New Orleans, Louisiana, October 1984.

[LLCK] T. Levitt, D. Lawton, D. Chelberg, and K. Koitzsch. Qualitative navigation of large scale space. to appear.

[LLCN87] T. Levitt, D. Lawton, D. Chelberg, and P. Nelson. Qualitative navigation. In *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, CA, February 1987.

[Mor80] H. Moravec. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover.* PhD thesis, Stanford University, September 1980.

[MS86] L. Matthies and S. Shafer. *Error Modeling in Stereo Navigation.* Technical Report CMU-CS-86-140, 1986.

[Pen85] A. Pentland. A new sense for depth of field. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 988–994, IJCAI-85, Los Angeles, California, August 1985.

[Pyl84] W. Pylyshyn. *Computation and Cognition: toward a Foundation for Cognitive Science.* MIT Press, Cambridge, Massachusetts, 1984.

[RA76] D. F. Rogers and J. A. Adams. *Mathematical Elements of Computer Graphics.* McGraw-Hill, 1976.

[SC82] R. N. Shepard and L. A. Cooper. *Mental Images and their Transformations.* MIT Press, 1982.

[SC85] R. Smith and P. Cheeseman. *On the Representation and Estimation of Spatial Uncertainty.* Technical Paper Grant ECS-8200615, September 1985.

[Sch84] H. Schone. Spatial orientation - the spatial control of behavior in animals and man. In R. Capranica, P. Marler, and N. Adler, editors, *Princeton Series in Neurobiology and Behavior*, 1984.

# Cooperative Methods For Road Tracking In Aerial Imagery

**David M. McKeown, Jr.**
**Jerry L. Denlinger**

***Digital Mapping Laboratory***
***Department of Computer Science***
***Carnegie Mellon University***
***Pittsburgh, PA. 15213***

## Abstract

This paper describes research in digital mapping and image understanding in the area of automated feature extraction from aerial imagery. We discuss a system for road tracking, ARF (**A** Road Follower), that uses multiple cooperative methods for extracting information about road location and structure from complex aerial imagery. This system is a multi-level architecture for image analysis that allows for cooperation among low-level processes and aggregation of information by high-level analysis components.

Two low-level methods have been implemented; road surface texture correlation and road edge following. Each low level method works independently to establish a model of the center line of the road, its width, and other local properties. Intermediate-level processes monitor the state of the low-level feature extraction methods and make evaluations concerning the success of each method. They also extract various road properties such as width changes, surface material changes, and overpasses. As a result of these evaluations one tracking method may be suspended due to apparent failure and restarted from the model generated by other successful trackers. Finally a high-level module generates a symbolic description of the road in terms of various attributes of the road such as center line, road width, surface material, overpasses, and an indication of potential vehicles on the road. This description is available in both map and image coordinate systems, and can be used to generate a textual description of the road.

## 1. Introduction

Road tracking in remotely sensed imagery has often been equated with linear feature extraction. The rational was that finding linear features in imagery either by region extraction or line finding was equavalent to finding roads. Further, it also worked for other types of linear features such as drainage, bridges and railroads. This view was appropriate considering the very low resolution imagery, LANDSAT I,II that was available at the time. Howe it is no longer appropriate given that the research community n as access to large scale, high resolution aerial mapping imagery allowing for the *structural analysis* of the road surface. In fact, for practical mapping applications such detailed analysis must be performed in order for a road extraction system to be a useful component in a digital mapping environment.

A second development is the emergence of computer vision systems that perform integration of image processing results, notably edge-based and region-based techniques, to generate more robust intermediate representations [5, 6, 2]. These systems can be contrasted with traditional image processing systems that exhibited a single line of sequential processing without much effort in the construction of intermediate intepretation structures or high-level analysis. However, even these innovative approaches exhibit single thread control structures even though they integrate multiple lines of sequential image processing.

ARF (**A** Road Follower), is the first system to utilize multiple independent control structures resulting in multiple lines of analysis and the generation of alternative intermediate representations. It is also unique in that it explicitely uses a cooperative method to allow for single point failures to be overcome by using an alternative (successful) tracking method. In this paper we report on:

1. An investigatation on the use of multiple sources of knowledge in low-level and intermediate level vision.
2. Measuring the effectivness of combining knowledge sources whose failure modalities are presumed to be independent.
3. A discussion of general purpose techniques for road tracking that would not make assumptions of scale, sensor, or road construction techniques.
4. The generation of symbolic descriptions of the road path, detectable events on the surface, and structural changes in the topology of the road.

In Section 2 we present a brief discussion of previous work in the automated extraction of roads from remotely sensed imagery. Section 3 describes the two low-level tracking methods. Section 4 focuses on the nature of cooperation in ARF. Sections 6 and 7 discuss some tracking examples and a set of experiments performed to establish performance of ARF run using individual trackers, and both trackers in cooperative mode. Finally, section 8 briefly discusses some future work.

## 2. Background and Previous Work

In this section we survey five research systems over the last ten years that were developed specifically for road tracking, or were developed as linear feature detectors and were demonstrated within the context of road delineation. For each system we outline several of the major assumptions based upon reading the published literature and give some subjective indication of performance based upon published examples. The types of "road knowledge" used by each method is described when appropriate.

### 2.1. Bajcsy and Tavakoli's road follower

Bajcsy and Tavakoli [1] used low resolution imagery from LANDSAT-1 where each pixel had a ground resolution of approximately 57 meters by 79 meters. Due to this low resolution, only very major roads of three or more lanes could be found. This system utilized the knowledge that roads are made of concrete or asphalt to directly determine the approximate intensity range to

This research was partially sponsored by the U.S. Army Topographic Laboratory Research Institute Fort Belvoir, VA, under Contract DACA72-84-C-0002 and by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 4976, monitored by the Air Force Avionics Laboratory Under Contract F33615-84-K-1520. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S Army Topographic Laboratory Research Institute, of the Defense Advanced Research Projects Agency, or the US Government.

expect in MSS band 2 of the LANDSAT-1 images. The paper does not say whether concrete and asphalt surfaces are processed separately or whether the results are merged at some point in the processing; possibly both materials look the same in LANDSAT-1 MSS band 2. It first performs a threshold operation, setting all points within some range of the expected road intensity to 1 and all other points to 0. Then it finds likely road points by scanning the entire image looking for points with the right intensity profile -- that is, points which match one of a number of templates indicating that there is a line point there. The templates are constructed so as to accept points that locally look like roads of width 1, 2 or 3 points, a total of 52 templates are used. Vertical, horizontal and diagonal roads and points where the road width changes by one point are accepted. It then grows the road by linking connected road points, using constraints on curvature and distance between road points. It then thins the road points so that the road is only one point wide. It then eliminates short segments as noise since short roads are not expected. It also labels intersections although the reason for doing this is not clear.

The program seems to find some of the roads in the imagery used although it also finds some spurious roads. The quality of the roads found is difficult to judge since photographs are not included in the paper. One obvious limitation of this work is that roads are expected to exhibit very specific spectral signatures. This assumption may work if the "right" multi spectral scanner (MSS) bands from LANDSAT-1 images are used but it will not work for images from many sources since different materials and lighting conditions can cause roads to be of any grey value. Also, it does not use much road-specific knowledge. The road knowledge used is knowledge about expected width for the given type of roads and imagery and knowledge about curvature and length. It is claimed that the program will find rivers, blood vessels and bubble chamber tracks as well as roads. This may be an indication of too little specialization for the task of finding roads since it is clear that rivers, blood vessels and bubble chamber tracks are similar only at the coarses level of description, ie., linear 2D shape. However, many of the features that distinguish roads from other linear objects are not visible at the resolution used so a general purpose line finder could be sufficient for the data.

### 2.2. SRI low resolution road tracker

The SRI low resolution road tracker [3, 4] used low resolution imagery (ground resolution is not reported). It looks for line features using some weak high level constraints on the shape of the road path. First an area of the image to search is designated, presumably by a user or a higher level program. It computes scores for several operators over the area designated, under the assumption that a combination of operators can do better than a single one. A distinction is made between object detection operators and object analysis operators. Object detection operators are binary operators which detect whether the object is definitely present or not. These operators are designed to detect only points which are very reliable, at the possible expense of a large number of omission errors. Object analysis operators are designed to give some quantitative measure of the quality of the object point once it is found. The object detection operators are first used to find all reliable road points. Then a cost array is generated for each object analysis operator. The cost arrays contain 0 for reliable road points and a function of the object analysis score for all other points. The function used can be designed to favor paths of a particular shape. A path optimization is then applied to each cost array and the path with the lowest score is chosen as the final road path.

Road knowledge used by the program is very weak. Some limitations in this approach are that both the road start and end points must be known at least approximately before tracking. In addition the results of the object analysis operators must be used in areas where roads have not been clearly detected in order to find a path between the points found by the object detection operators. However, the object analysis operators were designed to give reliable results only when the object is already known to exist. If it is really the case that the object analysis operators are only valid at points where the object is known to exist, it seems that they are completely superfluous.

### 2.3. Quam's road tracker

At SRI, Quam [12] used high resolution imagery, with ground resolution of approximately 1 to 3 meters. Quam's road tracker was part of the SRI road expert, the HAWKEYE system. It uses an algorithm based on correlation of the road surface pattern or intensity profile in the forward direction along the road. It must be given an initial starting point plus direction and width. This is expected to be supplied either by a user or a road finding program. It keeps a model of the road to use in finding the road points. The road model contains a surface model and a path model. The path model is a list of recent road points used for parabolic extrapolation of the path, the surface model is an array of intensity values sampled from the image in the direction perpendicular to the road. The surface model is allowed to change gradually as the road is followed.

The program first uses the path model to predict the position of the next road point one step ahead. It then extracts a cross-section from the image at the predicted point and performs cross-correlation with the road surface model to compute an error in the predicted position. The actual road position is computed from the correlation offset. If the correlation peak is poor, the program uses the path model to guess ahead another step and try to re-acquire the surface. It will continue to guess ahead until it finds a good match or until it has gone further than the length of the longest expected anomaly. If there are a large number of anomaly points, it then hypothesizes a surface change and extracts a new surface model. If it can follow the road successfully with the new surface model it continues, otherwise it quits.

The road knowledge used is the assumption that roads have a consistent surface wear pattern but with possible anomaly points, an assumption of constant width and the possibility of a change in surface material. However, it does not use any high-level knowledge about roads. For example, road width changes can cause problems due to gradually changing surface patterns that often occur at places where a new lane is added or one is deleted. Gradually changing surface patterns can also occur at intersections. This is a problem that does not occur with some of the techniques using lower resolution imagery. This makes it clear that although more information is available in higher resolution imagery, much more processing is required to take full advantage of it.

### 2.4. Nevatia and Babu's road finder

Nevatia and Babu [11, 10] used medium resolution imagery (ground resolution is not reported). Their program first runs an edge operator over the entire image to compute an edge magnitude and angle for each point. It then selects edge points based on three criteria:

1. The edge magnitude is greater than a fixed threshold.
2. The point is a local maximum in the direction perpendicular to the edge.
3. The edge angles of the neighboring pixels are within 30 degrees of the central pixel.

It then links the edge points together and fits piecewise linear approximations to the connected edges. Finally, it groups the line segments together into antiparallel pairs -- that is, pairs with opposite directions -- under the assumption that the road is of approximately uniform intensity and the background material is the same on both sides of the road.

The road information used by this method is the assumption of uniform road intensity and uniform background material, and assumptions of minimum length and straightness of roads. One problem is that it is difficult to group antiparallel lines properly, especially in the case where there are more than two lines. Another problem is that it is not always the case that the background material is the same on both sides of the road, for example, in the case where the road is one lane of a multilane highway and there is a median strip of another material separating it from the other lane. Finally, from a pragmatic standpoint, the edge test may be too strict to find many roads.

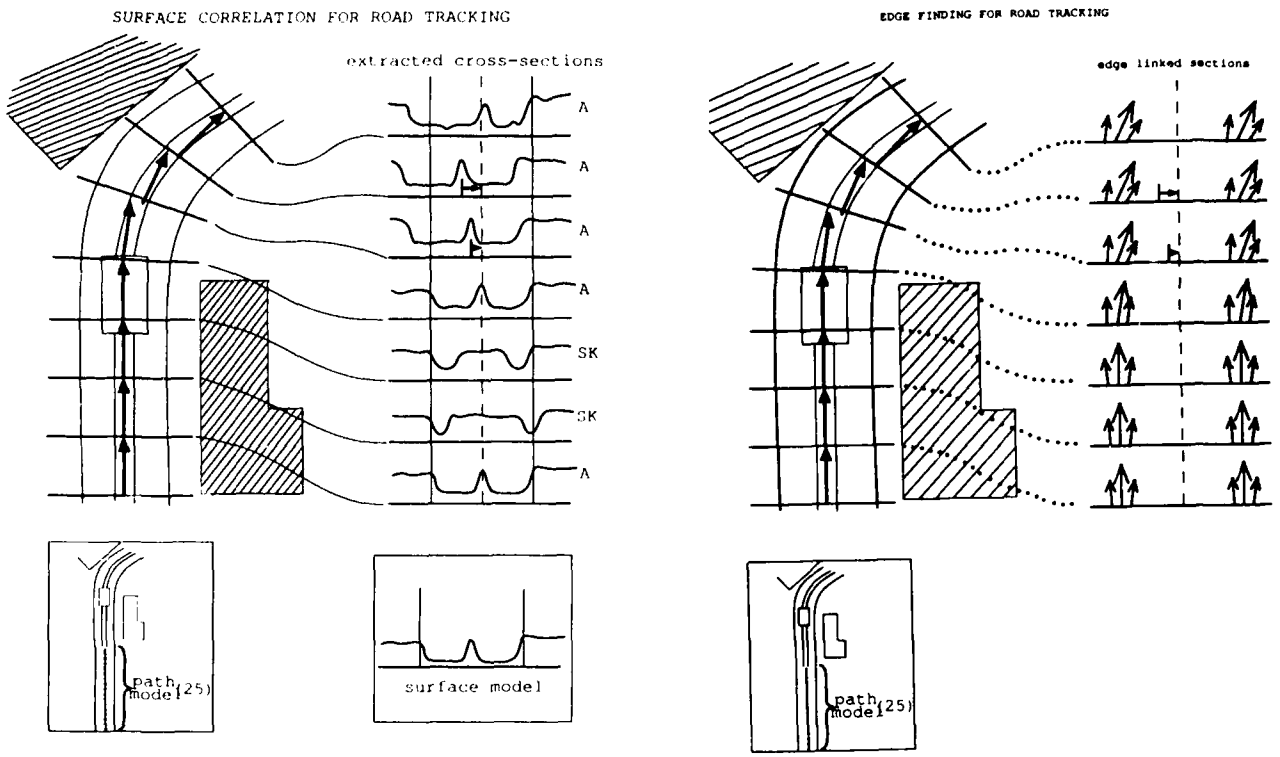



**Figure 3-1:** *Surface and Edge Tracking*

### 2.5. Kestner's road follower

Kestner [7] used medium resolution imagery where roads were on the average 3 or 4 pixels wide. It uses two methods to track roads -- a correlation follower and a region based follower. The program is started by a human or an automatic road finder with the road position and direction. The correlation follower looks in the direction of the expected road path for points with the expected intensity profile of the road against the background, that is, light on dark or dark on light. Where the correlation follower fails to find an acceptable path, a region based method is used to re-acquire the road. The region based method extracts a two-dimensional area from the image and searches the entire area for points with the expected intensity profile, marking each point with a score which indicates how well it matches the expected profile. It then eliminates all but the best points and links them together using constraints on road curvature to find the most likely road path. According to Kestner, the two correlation and region based trackers complement each other perfectly although it is not clear exactly how they are used together. A third method, called the binarizing method, is also described. This method first binarizes the section of the image which contains the road by setting all points within some range of the expected road intensity to 1 and all other points to 0. It then eliminates regions that are too wide to be part of the road and links the remaining thin regions together. It is not clear whether the binarizing method was implemented or how it was used in the system.

The primary road knowledge utilized concerns road shape and the assumption that roads will be of approximately constant intensity. An implicit assumption of this method is that the background is of approximately constant intensity. This assumption of constant intensity is a disadvantage since it cannot accomodate major surface changes or large changes in background intensity.

### 2.6. Summary of Previous Road Trackers

One can categorize road finder/followers into one of three major types; correlation trackers, region based followers, and edge linkers. All of these methods use a single local tracking strategy to find roads. Therefor a major problem with previous work is that if the method fails at some point it is very difficult to recover. Further, it is often difficult to automatically *recognize* when the tracking method has failed since the local nature of these methods assumes that the local maximum, no matter how poor, is its best guess for the position of the road. As a side note, it is difficult to compare the performance of the different trackers against each other since they all use different resolution imagery from different sources and there is no consistent method for reporting results.

We believe that the main way to improve road tracking is to develop multiple methods to extract more information about the road and to use cooperative techniques to allow a failed method to be detected and restarted using an alternative technique. In the following section we describe two tracking methods in sufficient detail that others should be able to implement similar road trackers. In section 4 we discuss the cooperative architecture within which these trackers are imbedded, and the use of feature specific detectors for road events.

## 3. Road Tracking Using Surface and Edge Models

In this section we describe the two tracking methods currently used by ARF, a surface correlation tracker and a road edge tracker. Each tracker makes assumptions and uses road specific knowledge in its image analysis. The surface tracker makes the following assumptions:

- a constant width
- a surface intensity profile which changes either very gradually or suddenly
- a path that is well modelled locally by a parabola
- slowly changing direction

The edge tracker makes the following assumptions:

- road edges will be fairly straight

- a path that is well modelled locally by a parabola
- slowly changing direction

Figure 3-1 gives an pictoral description of each of the tracking methods. These methods will be described in detail in the following sections.

### 3.1. Correlation-Based Road Tracking

This section describes a correlation-based road tracker for medium to high resolution images. It uses techniques first described in [12] with several improvements. All correlation-based techniques are based on the assumption that there is a discernable intensity pattern or texture on the road surface. For example, there is usually a light colored line down the middle of multiple-lane roads and there are usually darker wear patterns in the lanes themselves. We do not look for any such specific pattern but rather look for a pattern that is known to be on the road at some starting point. Provisions for slowly changing road surfaces and for sudden changes in road material (from asphalt to concrete, for example) are made.

The correlation tracker takes as input parameters the starting coordinates of the road, its initial direction and its width. In normal operation these parameters will be supplied interactively by a human or provided by a higher level program, perhaps by using information in a database to determine where a road might exist or by using some sort of road finding operator. A road is followed by extracting a cross-section perpendicular to the road from the image at points predicted by a road trajectory model and using cross-correlation with a cross-section model to determine the actual position of the road. If the correspondence is poor, or if the offset (the shift between cross-sections) is greater than expected, we continue guessing the road position until we find a good correspondence or we have gone so far that it is unlikely we will re-acquire the road. In the latter case, we assume the road surface has changed and attempt to find a new surface model.

The cross-section model is an array of intensity values taken from the image. The trajectory model is the coefficients of a parabola fitted to the most recent points on the road.

#### 3.1.1. Prediction: Taking A Small Step Forward

We predict the position of the road one step ahead by fitting a parabola to the most recent road points. We store a list of about twenty road coordinates to use for curve fitting. In addition to the points themselves we also store a correlation score, which is a relative measure of the quality of the cross-correlation, for each point. When we fit a curve to the points, we ignore those with the worst scores to get a correspondence with higher confidence.

In determining the number of points to use for correlation, several things must be taken into consideration. In order to get reasonable accuracy, we need to use enough points so that the fit is not too dependent on any one point. However, we cannot use too many points because the road may not be well suited for modeling with a parabola over long distances. In addition, the maximum number of points we can use for curve-fitting is dependent on the size of the steps we take since the number of steps that can be modeled well by a parabola will be smaller for larger steps. Thus, for greater accuracy in trajectory prediction we need to take smaller steps. However, with very short steps the acceptable correlation shift between successive cross-sections (corresponding to a maximum acceptable change in road direction over a small distance) becomes very small and nearly impossible to achieve. The selection of the best number of points to use is therefore a tradeoff between accuracy of road prediction and confidence in the road positions found.

To take a single step along the road, we use the following eight steps, which are identical in form to those described in [12]. The initial trajectory model is generated by extrapolating backwards along the initial angle given and storing the necessary number of points. In this way, we start out looking straight ahead without otherwise treating the first points as a special case. The initial cross-section model is created by taking the average of several cross-sections extrapolated forward from the initial point.

[ONE]: Extrapolate the position of the road one step forward.

We represent the road path parametrically as two separate functions x(t) and y(t) where t is the total length in steps that we have traversed along the road's path. We use multiple regression [13] with t and $t^2$ as the independent variables to fit parabolas to x(t) and y(t), getting approximate functions:

$$x'(t) = a + bt + ct^2$$

$$y'(t) = d + et + ft^2$$

It can be shown, by eliminating t from these equations, that the fitted curve is a parabola in the x-y plane as well as in the x-t and y-t planes. Further, the parabola in the x-y plane can have arbitrary orientation, having an equation of the form:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

We do not actually use this equation since the parametric form is more convenient for our purposes.

The position of the road is predicted by computing x'(t+1) and y'(t+1). Since this does not guarantee equal step sizes, due to nonlinearity of the fitted curves, we must adjust the predicted position so that the length of the step taken is equal to the step size we intended. The problem is that we are using t to predict position and t has only an implicit, and not necessarily constant, relation to the length traversed along the road. The error for any point is normally fairly small, the main concern is that these errors do not accumulate. We first find the tangent to the fitted parabola near the current point by taking the slope of the line connecting the fitted points for t and t+1. This gives an average tangent between the two points. We then project the previous road point onto the tangent line by finding the intersection of the tangent line and the line which is perpendicular to the tangent and goes through the previous point. The predicted road point is then found by taking one step forward along the tangent line from this intersection.

[TWO]: Extract road cross section.

We take a cross section perpendicular to the road at the point predicted in step [ONE], using a linear interpolation to get multiple sample points per pixel. To allow for cross-correlation, we take a cross-section wider than the road itself. We take a straight cross-section as opposed to a circular one as suggested for low resolution images in [7] both for simplicity and because it is not clear what radius of curvature to use or how to match circular cross-sections when we are looking for a pattern on the road itself.

The linear interpolation used is a weighted average of the four pixels around the floating point image coordinates desired. We imagine the pixels as being squares with their integer coordinates representing the point in the center of the square. To determine the weight used for each neighboring pixel, we draw a square of area one pixel around the point requested. The weight used for each pixel is the area of overlap between the square and the pixel. The advantage of this interpolation method over non-linear methods, such as that described in [14], is speed. We need a lot of points to get a good correlation and we can perform several linear interpolations in the time it takes to do a single non-linear interpolation.

[THREE]: Cross-correlation.

We do a simple valley finding correlation to determine the best match between the cross-section model and the new cross-section. We look for the offset that gives the smallest value for the sum of the squares of the differences between corresponding elements. We compare the cross-section from the model with a window on the new cross-section -- shifting the window to get the score for different offsets. The correlation is done only over the width of the road. To reduce the number of sums we need to compute, we stop as soon as we have a valley. In doing this, we are assuming that the point we are looking for will be close to the center -- using the first valley we find reduces the chances of finding a good correspondence that is not what we are looking for (for example, a different lane of the road that happens to yield a better correspondence).

[FOUR]: Generate a mask indicating the positions of noise elements.

Using the correlation computed in step [THREE], we create a mask of the elements which differ from their corresponding points in the surface model by a significant amount. These are noise elements which will be ignored in a second, more accurate correlation.

[FIVE]: Re-correlate over the unmasked elements.

To get a better match we re-correlate using only the unmasked elements. The basic correlation used is the same as in step [THREE]. To get a sub-element match we fit a parabola to the three points closest to the valley and take its minimum. We check several things to see if the correspondence is good enough. We check the magnitude of the offset of the correspondence, the sum of the squares of the differences (correlation score) and the number of points marked as noise points. By using three different measures, we need not be so strict on any single one. If any one of these is too large, we will not accept the correspondence. If the correspondence is not good, we guess the position of the road on successive steps until we find a good correlation and then backtrack to output the road points.

[SIX]: Update the cross-section model.

If the correspondence was good, we create a new surface cross-section model from a weighted average of the old surface model and the matched cross-section. This essentially results in an exponentially decaying model. A large weight is used for the model and a small weight for the new cross-section to prevent the model from changing too rapidly.

[SEVEN]: Adjust the position of the road center.

Using the offset from the correlation, we generate the road point. We store this in the output and also add it to the list of points used to predict the road position, and delete the oldest point from the list.

[EIGHT]: Anomaly detection.

Anomalies are found in the same way the mask elements were found in step [FOUR] except a greater difference is tolerated. We are more restrictive for detecting anomalies than we are for ignoring noise and random fluctuations. In the current version, the step size is greater than one pixel so we must do multiple lines of anomaly detection for each step.

#### 3.1.2. Following, Guessing and Backtracking

At each step along the road, we check if the correspondence is acceptable (in step [FIVE]). If it is, we update the surface and trajectory models and try to advance another step. If correspondence is poor we must figure out why and try to find where the road really is. Poor correspondence can be caused by two things:

1. There is something occluding most or all of the road. This could be a vehicle, an overpass, a shadow, etc.
2. The road surface has changed. For example, if we are going over a bridge and the surface has chanced from asphalt to concrete.

We treat these two cases differently. In the first case, we skip over the occluded section until we can re-acquire the road. We use the backtracking as suggested in [12] to avoid outputting false anomalies. In the second case, we construct a new cross section model and continue following.

Believing that occlusions are more common than surface changes, we check for occlusions first. We look ahead along the extrapolated road position and try to find a cross-section with a high correlation with the current model. When we are guessing the position of the road, we allow a greater absolute deviation from the expected position if we have been guessing for a while (that is, the acceptable position of the road spreads out as we guess for a greater distance). A linear function of the number of guessed steps is added to the normal acceptable offset to get the offset we will actually tolerate. If we find a good correlation along the guessed trajectory, we test to make sure we have really found the road by requiring that more than one consecutive cross-section give a reasonable match. If we have been guessing the road for only a few steps, we only require that one point give good correspondence. If we find a good correspondence, we backtrack and do road track and anomaly output for the points we skipped over, starting from the last successful point. We then continue following the road from the newly found position.

When we are guessing, if we find a point with good correspondence we add it to a temporary road model (for both the road surface and the trajectory). In subsequent extrapolations we use the temporary models under the assumption that we are actually following the road. This allows us to guess the road position based on all the information we have and makes backtracking easier if we decide we have found the road. If we eventually decide the points added to the temporary road model are really on the road, we backtrack, adding the intermediate points to the permanent model.

If we decide the points are not good, we throw out the temporary model and make a new temporary model out of the old permanent model. When we create the new temporary model, we do not go all the way back to where we started guessing, we just get rid of the new points we added and continue from where we are.

If we have not found a reasonable correspondence with the current model by extrapolating forward we presume the road surface has changed. We then try to start a new model by retrieving a new cross-section at the last successful point to serve as the model. If we find a good correspondence with the new model, we backtrack over any points we skipped, then begin following again. If we do not find a good correspondence, we use the cross-section where the new correspondence failed as the model and try again. We continue doing this until we find a good correspondence or we have gone so far that it is unlikely that the road will be re-acquired.

### 3.2. Edge-Based Road Tracking

The edge tracker is based on the road finding method of Nevatia and Babu. It tracks the edges of the road by linking points with high gradient and orientation in the direction expected for the road. The gradient and orientation are computed by a 5 X 5 Sobel gradient operator. We use the Sobel gradient rather than the Nevatia-Babu operator because we found that the 12 value grain of the Nevatia-Babu operator was too course for accurate angle comparisons.

During tracking, the position of the next road point is predicted by parabolic extrapolation from the recent path the same way as is done in the surface tracker. The Sobel is then computed for the points along the line perpendicular to the road direction at the predicted point. A score based on the weighted sum of several component scores is computed from the Sobel values for each point. Each of the components is a linear function between 0.0 and 1.0 in the range between some minimum and maximum, 0.0 outside the range at one end and 1.0 outside the range at the other end. The component scores are edge strength, orientation, difference in magnitude from each of the neighboring points and difference in angle from each of the neighboring points. The edge strength score is high for edges of greater magnitude, the orientation score is high for angles of the expected orientation. The difference in edge magnitude score is high for larger differences, selecting for points whose neighbor points have relatively small edge strengths. The difference in angle gives high score to points whose neighbors have similar angles, selecting for points in areas of consistently linear texture.

The final score for each point is taken as the sum of all of the components. The point with the highest score is then chosen as the edge point. If no points have a score higher than some minimum threshold, no point is selected as the edge point. If edges are found on both sides, the center line point is marked as the point in the middle. If only one edge is found, it is assumed that the width has remained constant and the center line point is marked as half of the width from the one edge found. If no edges are found, guessing ahead is done the same way as for the surface tracker. As with the surface tracker, if guessing ahead goes too far without finding a good point, the tracker quits.

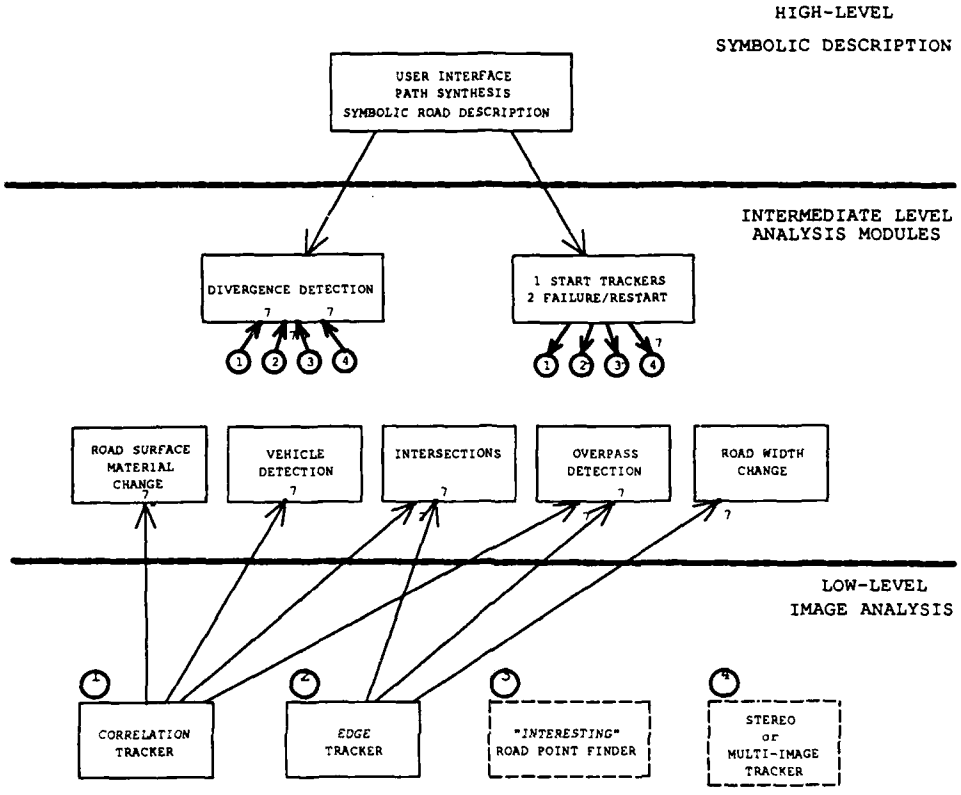



**Figure 4-1:** ARF System Organization

To decrease the possibility of finding an edge point that has a high score locally but is not part of the road edge, edge linking is done between points where possible. We use the same method as Nevatia and Babu, looking ahead in the direction of the edge and accepting the next point if it is a maximum in edge magnitude and has the correct direction.

## 4. Cooperative Road Tracking

In this section we discuss the organization of ARF, an Automatic Road Follower. ARF is organized into three processing and analysis levels. The low-level is composed of two independent low-level road trackers that generate an estimation of the center line of the road and produce various tracker dependent features such as width, surface material change, and anomaly detection. These trackers have been described in detail in Sections 3.1 and 3.2. The intermediate-level is composed of several modules that detect and report road features such as overpasses, road width changes and vehicle detection. The high-level provides overall control and user interface. It detects situations where one low-level tracker must be restarted from the other and allows users to interact to start or restart the system. Figure 4-1 gives a description of the overall organization of the cooperative road tracker system.

The basic control cycle in ARF is to invoke each of the road trackers independently and allow them to track asynchonously until each has generated a step forward. This step might require several advance, fail, guess, and prediction steps within each of the road trackers as described in Section 3. Once both trackers have completed a step forward, feature detection is performed by invoking each of the intermediate level detecters with access to the internal path model of both trackers. These detector modules and the type of state information that they require is described in Section 5. Finally the control queries whether a path divergence has occured. Such a divergence is currently a difference of 7.5 meters. If no divergence is detected the basic control cycle continues.

Upon detection of a divergence, the high-level invokes each tracker to advance 5 additional units. The path history and confidence values are used to evaluate the relative goodness of each tracker, both in terms of its history, and in terms of the 5 additional units. The high-level can then decide to terminate the overall run, if the confidence in both trackers is sufficiently low, or can decide to restart one tracker from the path model of the other.

The process of restarting is made straightforward since the internal data structure representation of the path history for each tracker is identical. Thus an edge tracker path history can be loaded directly from the surface tracker, and edge values and confidence can be computed without search. Similarily the correlation tracker can be forced to assume a new path model, and rebuild its weighted decay cross section.

## 5. Detection of Road Features

There are six types of road features which ARF attempts to detect and delimit -- intersections, width changes, overpasses, surface material changes, vehicles, and occlusions. Any of these conditions can cause simple tracking schemes to fail. Multiple conditions, such as occlusions on a sharp curve, width changes followed by a surface material change require multiple sources of road estimates in order to be robust. Each of the intermediate level feature detectors has access to each of the trackers internal road history data structure. This data structure contains an entry for each path point tracked on the road including center position, match confidence, edge confidence, width estimate, left/right edge track, and model history. The results of each feature detector is combined with the original tracker information to form a composite path model. For each point in the composite path model we store

- edge quality
- correlation quality
- road width
- number of anomaly points
- location of anomaly points in the cross section (mean, std. dev.)
- difference between edge and surface centers
- type of any feature detected (overpass, vehicle, etc.)
- average surface intensity
- intensity of anomaly points (mean, std. dev.)

In the following sections we summarize how each feature detector determines that a feature event has occured, how it is verified, and what actions (if any) should be taken by the high level control. The major problem for most feature detectors is in the generation of hypotheses about continuous features that occur over several path points only using pointwise information as above. There role is to survey and combine the noisy pointwise into the occurance of a discrete event. A second problem is in the determinization of the actual location of the feature. For example, in the case of width changes, the actual feature occurs, usually slowly over many road points. This localization is important since we rely on reasonably accurate positions for overpasses and intersections in order to skip over them.

### 5.1. Intersections

To detect intersections, we detect cases where there are several consecutive road points with bad edges and possibly with poor correlation scores. We also expect no significant change in average pixel value over the cross-section. On roads in residential and urban areas, we might expect intersections to appear at fairly regular intervals. The history then must contain a measure of the quality of the edges and a measure of the quality of the correlation scores. Verification can be accomplished by adding the intersection location to an agenda of possible new road starting points and invoking the tracker using the intersection width and direction[1]. To skip over, look ahead past the predicted width of the intersecting road. It is generally not necessary to flush portions of the surface model.

### 5.2. Road width change

The detection of road width changes requires the integration of several clues. During a width change we expect to find good edges, possibly poor surface correlation, and there should be anomaly points along one side of the road. We use the path history to determine the number of anomaly points for each road point and their location on the cross-section. In addition, there should be some disagreement between edge center and correlation center if both trackers give good scores. Significant width changes can be detected directly if a long enough history of the edge points is maintained. Since width changes often occur on highways near overpasses, if an overpass has already been detected we increase the possibility of width change. These types of interactions argue for a tightly coupled path history accessible to all feature detectors. In this case, verification is not necessary since the width can be detected directly.

The most important implication of a road width change is the eventual breakdown of the surface tracking model because its correlation cross section no longer models the actual road cross section. This is exactly what causes the generation of anomaly points along one edge, usually the edge that is expanding or contracting. As a result of a road width change the model width is updated and it is necessary to re-acquire a surface model using the path model of the edge tracker.

### 5.3. Overpasses

Overpasses exhibit properties similar to intersections at the low-level path history. The major distinguishing feature is that we should expect some changes in the average surface intensity, more so than in an intersection. Generally we detect lack of consistent road edges conincident with a series of bad surface correlation scores. Overpasses are likely to have shadows that precede or follow the previously mentioned features, and these shadows are usually detected as anomalies. Verification can be perfromed by finding leading and trailing edges of the overpass, usually also detected with anomalies..

### 5.4. Surface material changes

Surface material changes are relatively easy to detect, since the pavement changes causing a large, abrupt change in average surface intensity along with poor surface correlation but possibly good edges. Often surface material changes coincide with bridge decks, and two closely spaced changes can be detected. Short changes due to patching are detected as anomalies. In this case the path history is re-acquired using several consecutive matching cross-sections.

### 5.5. Vehicles

Vehicles can be detected by looking for patches of anomaly points of approximately uniform intensity. Since the path history information stores the number of anomaly points, their average intensity, and their location in the cross-section, the vehicles feature detector performs simple grouping.. Since vehicles can appear at any point on a road but there should be some rules about where they can be with respect to each other and rules about which lanes they are likely to be in.

### 5.6. Occlusions

Occlusions are detected initially by looking for irregular patches of anomalies along the side of the road. Such irregular patches can be caused by buildings and trees. They are too long and irregular to be vehicles, tend to not cause bad correlations but may coincide with intermittent edges on one side of the road.,

## 6. Some Road Tracking Examples

Photographs 6-1, 6-2, and 6-3 are three typical examples of the ARF cooperative road tracker. Photographs 6-1 and 6-2 are continuations of the same tracker run on a limited access highway with a ground sample of about 1 meter per pixel. Figure 6-3 shows a similar road at more about 3.5 meters per pixel.

ARF generates a real-time display with four display windows. The 'edge tracker' window shows the independent progress of the edge tracker including centerline, edge points and predictions and guesses. The same information is displayed in the 'surface tracker' window, except that only the anomalies replace the edge points. The 'cooperative output' window displays the result of the high-level integration of centerline paths and the superposition of road feature events. The text window gives a running interpretation of the feature symbols superimposed on the cooperative output. The three image windows scroll independently as edge and surface tracking proceed. The cooperative output usually lags the output of the trackers since it often must wait for delayed events from the feature detectors.

[1]This really becomes a road finding problem, not addressed in this paper, but addressed in related work on road network generation

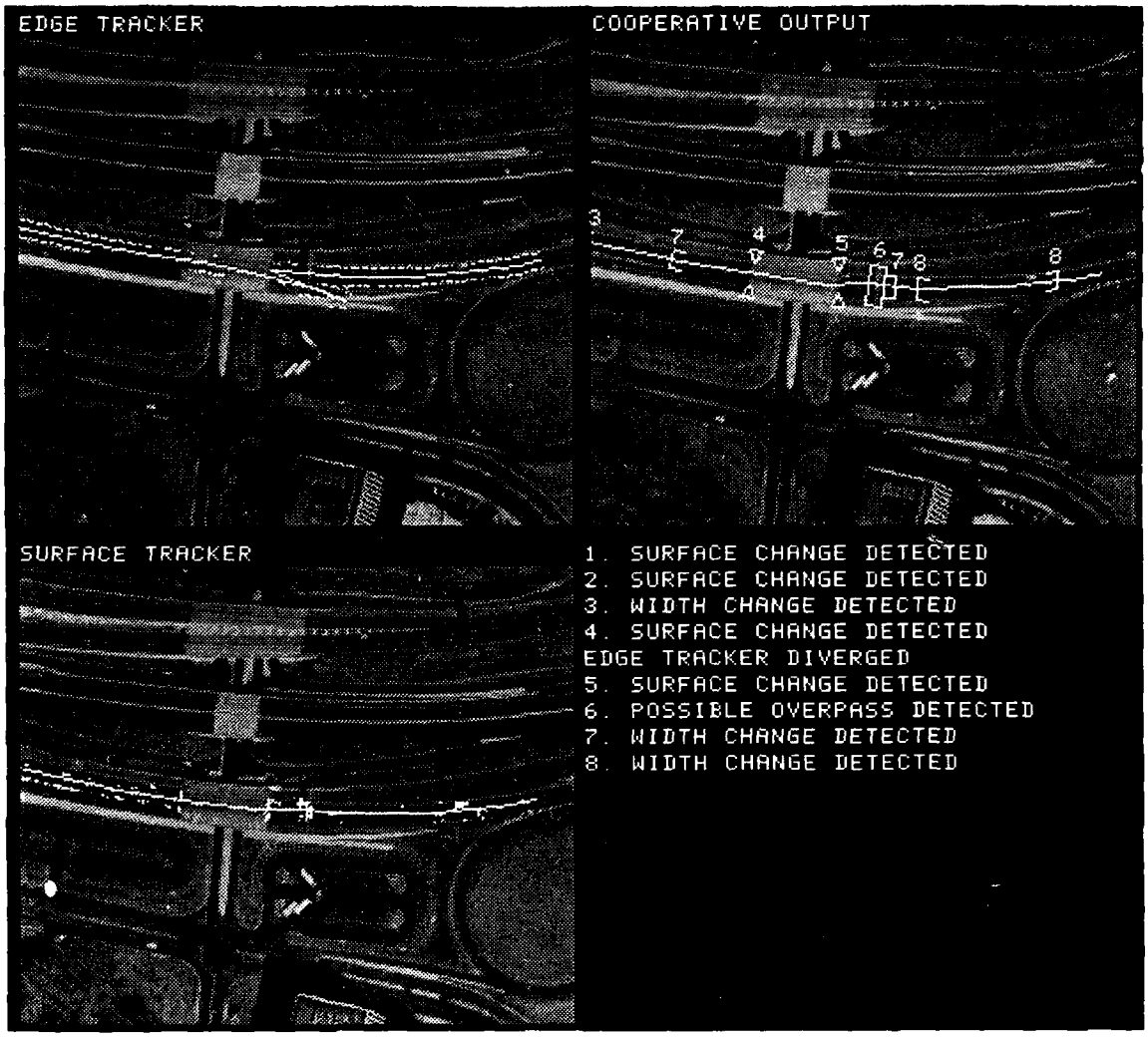



**Figure 6-1:** PHOTO 1

*Photographs* 6-1 and 6-2 show several examples of road width changes, surface material changes and overpass detection. Anomaly detection is particularly visible in photograph 6-2 in the area of the overpass. While initially this area is marked as an anomaly, the cooperative output correctly identifies it as an overpass (12).

Photograph 6-3 shows an example of nearly complete failure by the edge tracker in a low resolution road at the point where several roads merge and the road direction changes. However, the surface tracker correctly tracks the direction change and the cooperative output reflects the use of each trackers confidence scores to correctly weight the final output.

### 6.1. Textual Description From Road Extraction

Figure 6-4 illustrates the symbolic descriptions that are produced by ARF during a tracking session. All imagery used by ARF is maintained in the MAPS database [8, 9]. The MAPS database provides a digital elevation model and a camera model that allows for image-to-map and map-to-image correspondence. ARF uses these image-to-map correspondence equations to transform sub-pixel road centerline positions into geographic coordinates, and then uses the geographic coordinate to index into a digital elevation databae to calculate an interpolated terrain elevation. This capability allows for reasoning about image features in metric distances as opposed to pixel units and is used to represent distances in each of the intermediate-level feature detectors..

## 7. Performance Evaluation

Our input data consists primarily of aerial photographs of the Washington D.C. area from two different sources. We selected 35 roads for tuning and initial debugging and 35 for testing, attempting to select fairly difficult data so as to allow for future improvements to be reflected in tests on the same input. We adjusted the parameters of the program to obtain the best overall performance (measured subjectively) when run on the tuning roads and then ran it on the test roads also. We intend to run future versions of ARF on this input and use the numbers given above as benchmarks. Performance data for tuning and test roads are given in Figures 7-1, 7-3, and 7-5 for the "training" data and Figures 7-2, 7-4, and 7-6 for the blind "test" data. For each table road *Width* and *Length* are in pixels [meters]. *Time* given are system and user time, respectively, in seconds. Under *Subjective Reason for Stopping*, **A** indicates automatic stopping, **M** indicates [illegible] the program was stopped manually by the operator.

When ARF does not follow a road to the edge of the image, it usually fails for one of the following reasons:

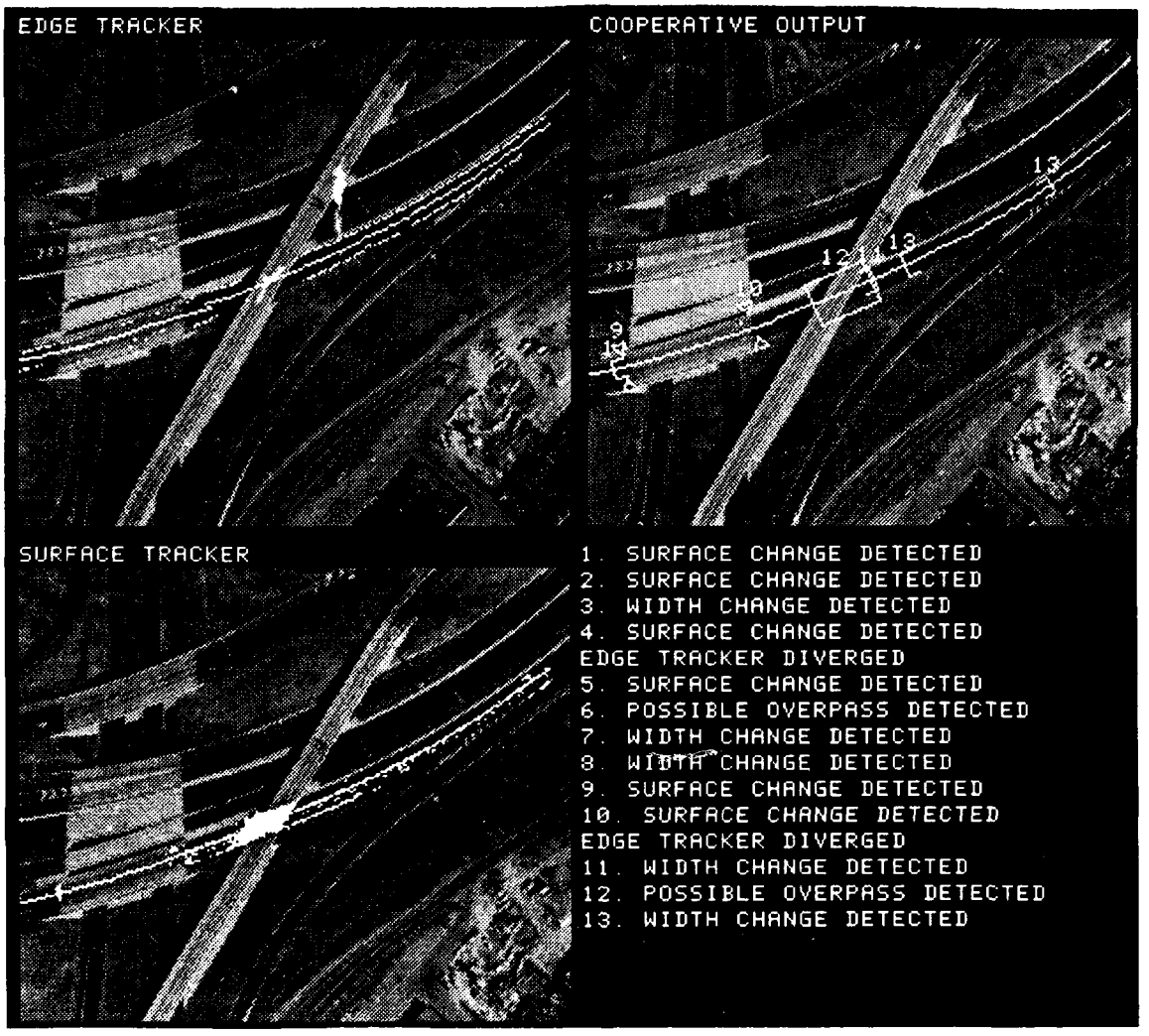



**Figure 6-2:** PHOTO 2

- The curvature of the road is too great. This is usually only a problem when there is some sort of obstruction on the road at the curve.
- The material beside the road is the same intensity as the road itself. This becomes a problem if there is little texture on the road. This problem also occurs at intersections where two roads of the same material meet and in fuzzy images.
- A severe and sudden road width change. This is fairly uncommon on highways in urban scenes. Addition of a lane does not usually cause problems but losing a lane causes false anomaly output in some cases and complete failure due to a large number of detected noise pixels in other cases.
- The road does not curve smoothly or its path is not suitable for modelling with a parabola for other reasons (for example it might be S shaped, curving left and then immediately curving right).
- There is no consistent pattern on the road surface. That is to say that successive cross-sections along the road path do not show a high degree of correlation. This is rare but it does occur.

Additionally, in some cases the road path generated is not very smooth or does not stay on the center of the road. This usually happens in fuzzy images or on roads with little texture.

Speed is usually between 7 and 14 pixels of road length per second of CPU time on a VAX 11-785. Timings depend mostly on the road width, the number of anomalies and the amount of guessing and backtracking necessary. About 35 percent of program time is spent in pixel access and interpolation.

Figure 7-7 gives the relative performance of the each tracker vs. the other method and vs. the combined tracker. These results indicate that the surface tracker is of significantly higher quality than the edge tracker. The surface tracker produces a longer track for nearly all of the runs.

It is important to note that the combined tracker is better than either tracker alone in a significant number of cases. Although the individual trackers are better than the combined tracker in some cases, improvements to the mechanism which measures the quality of the path could increase the overall quality of the combined tracker.

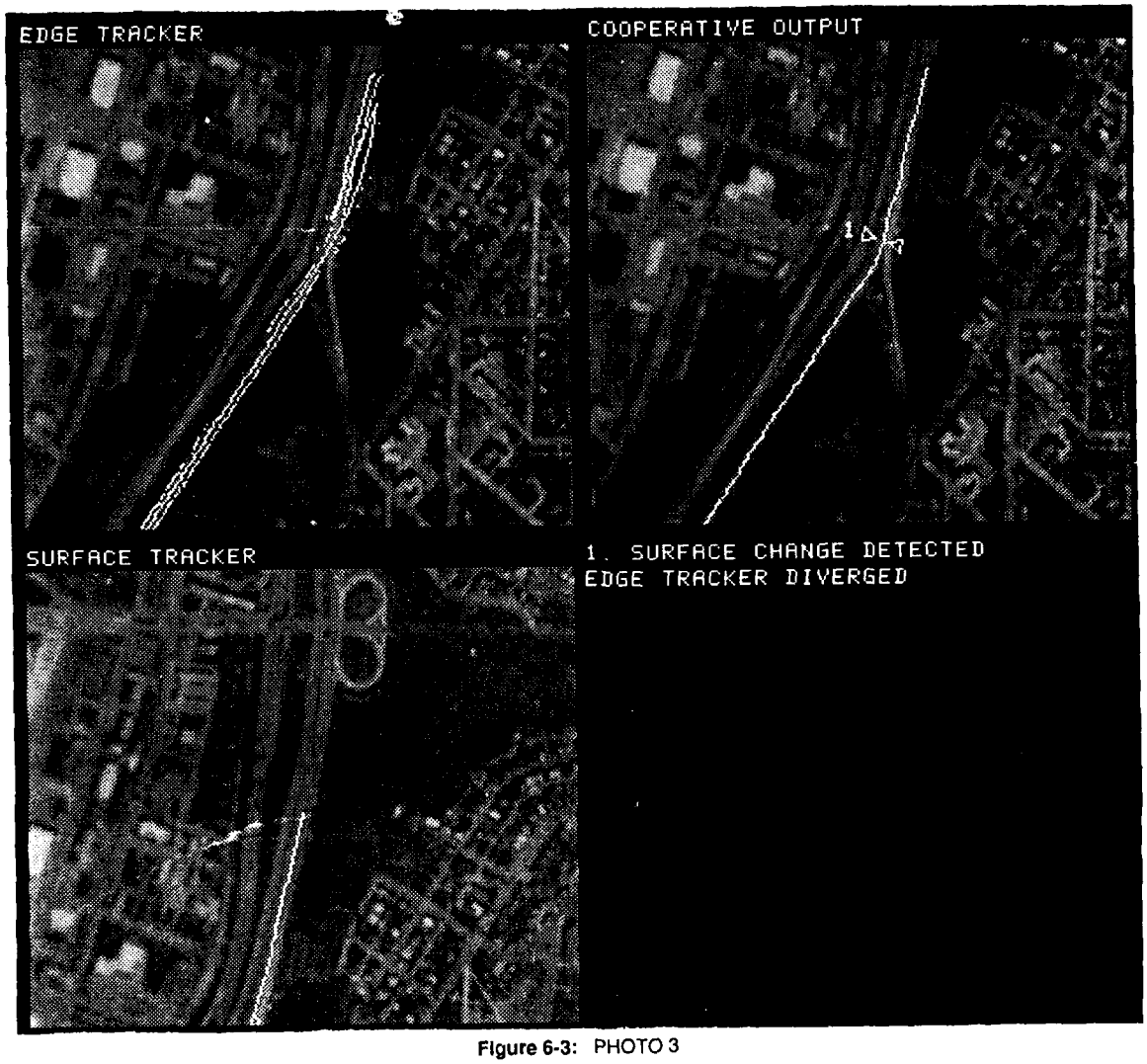



**Figure 6-3:** PHOTO 3

## 8. Conclusions

In this paper we have presented a complete system for road tracking and feature detection in high resolution aerial imagery. The system has been tested on a large number and variety of complex aerial imagery. We believe that its performance would in many cases justify its use as an interactive assistant for road network delineation. The quality and reliability derives from the use of two, independent methods for road tracking, organized into a cooperative system architecture that tolerates failures and allows for automatic restart.

There are many avenues for future work. First and foremost we are investigating the use of an automatic road starting point finder so that we can relax the requirement that ARF be given an initial starting point, road width, and direction. Second, we believe that other tracking methods could be integrated into the ARF system, such as the use of multi-temporal imagery, or stereo pairs. What remains to be seen is whether additional information from other sources actually improves overall system performance.

Finally, ARF should be viewed in a larger context of road network extraction. That is, at many points where ARF fails a road finder could be invoked to attempt to locate new plausible starting points which would allow for continuation. ARF generates areas such as overpasses or surface material changes which should also become the focus of road finding. Road network extraction then becomes the integration of various search strategies for invocation of road finding and road tracking.

## 9. Bibliography

[1] Ruzena Bajcsy and Mohamad Tavakoli.
Computer Recognition of Roads from Satellite Pictures.
*IEEE Transactions on Systems, Man, and Cybernetics* vol. SMC-6(9):623-637, September, 1976.

[2] Fau, P., and Hanson, A.J.
Using Generic Geometric Models for Intelligent Shape Extraction.
In *Proceedings of Sixth National Conference on Artificial Intelligence*, pages 706-711. AAAI, July, 1987.

[3] M. A. Fischler, et. al.
The SRI Road Expert: An Overview.
*Proceedings: Image Understanding Workshop* :pp. 13-19, November, 1978.

[4] M. A. Fischler, J. M. Tenenbaum, and H. C. Wolf.
Detection of Roads and Linear Structures in Aerial Imagery by Computer.

```
Road 32 begins at                    lat N 38 50 51 (696)   lon W 77 3 32 (889)
                                     (elevation: 30.90 meters [101.38 ft])
    L-intersection 0 begins at       lat N 38 50 51 (665)   lon W 77 3 31 (614)
                                     (elevation: 31.66 meters [103.87 ft])
    L-intersection 0 ends at         lat N 38 50 51 (665)   lon W 77 3 31 (614)
                                     (elevation: 31.66 meters [103.87 ft])
    Total distance over last L-intersection is 3.12 meters [10.23 ft]
    R-intersection 1 begins at       lat N 38 50 51 (646)   lon W 77 3 31 (401)
                                     (elevation: 31.66 meters [103.87 ft])
    R-intersection 1 ends at         lat N 38 50 51 (646)   lon W 77 3 31 (401)
                                     (elevation: 31.66 meters [103.87 ft])
    Total distance over last R-intersection is 2.28 meters [7.47 ft]
Road 32 ends at                      lat N 38 50 51 (639)   lon W 77 3 29 (859)
                                     (elevation: 31.64 meters [103.81 ft])
    Total distance is 48.58 meters [159.38 ft]
    Total Elevation change is 0.740000 meters [2.43 ft]

Road 38 begins at                    lat N 38 50 54 (224)   lon W 77 3 33 (121)
                                     (elevation: 36.08 meters [118.37 ft])
    L-intersection 2 begins at       lat N 38 50 54 (266)   lon W 77 3 33 (67)
                                     (elevation: 36.29 meters [119.06 ft])
    L-intersection 2 ends at         lat N 38 50 54 (266)   lon W 77 3 33 (67)
                                     (elevation: 36.29 meters [119.06 ft])
    Total distance over last L-intersection is 4.38 meters [14.37 ft]
    L-intersection 3 begins at       lat N 38 50 54 (242)   lon W 77 3 31 (73)
                                     (elevation: 35.38 meters [116.08 ft])
    L-intersection 3 ends at         lat N 38 50 54 (242)   lon W 77 3 31 (73)
                                     (elevation: 35.38 meters [116.08 ft])
    Total distance over last L-intersection is 7.15 meters [23.46 ft]
    width 8 begins at                lat N 38 50 54 (228)   lon W 77 3 30 (249)
                                     (elevation: 34.27 meters [112.43 ft])
    X-intersection 4 begins at       lat N 38 50 54 (208)   lon W 77 3 30 (143)
                                     (elevation: 34.04 meters [111.68 ft])
    road-split 5 at                  lat N 38 50 54 (230)   lon W 77 3 30 (36)
                                     (elevation: 33.95 meters [111.38 ft])
    X-intersection 7 begins at       lat N 38 50 54 (254)   lon W 77 3 29 (717)
                                     (elevation: 33.56 meters [110.10 ft])
    X-intersection 4 ends at         lat N 38 50 54 (208)   lon W 77 3 30 (143)
                                     (elevation: 34.04 meters [111.68 ft])
    Total distance over last X-intersection is 7.08 meters [23.22 ft]
    X-intersection 7 ends at         lat N 38 50 54 (254)   lon W 77 3 29 (717)
                                     (elevation: 33.56 meters [110.10 ft])
    Total distance over last X-intersection is 3.37 meters [11.04 ft]
    width 8 ends at                  lat N 38 50 54 (228)   lon W 77 3 30 (249)
                                     (elevation: 34.27 meters [112.43 ft])
    Total distance over last width is 13.07 meters [42.87 ft]
    Changed from 3.85 meters [12.64 ft] to 5.17 meters [16.96 ft]
    R-intersection 6 begins at       lat N 38 50 54 (277)   lon W 77 3 27 (137)
                                     (elevation: 31.49 meters [103.31 ft])
    R-intersection 6 ends at         lat N 38 50 54 (277)   lon W 77 3 27 (137)
                                     (elevation: 31.49 meters [103.31 ft])
    Total distance over last R-intersection is 4.83 meters [15.83 ft]
Road 38 ends at                      lat N 38 50 54 (471)   lon W 77 3 24 (101)
                                     (elevation: 31.20 meters [102.36 ft])
    Total distance is 147.91 meters [485.27 ft]
    Total Elevation change is -4.880000 meters [-16.01 ft]
```

**Figure 6-4:** Textual Description From ARF


*Proceedings: Image Understanding Workshop* :pp. 87-100, November, 1979.

[5] Huertas, A. and Nevatia, R.
Detection of buildings in aerial images using shape and shadows.
In *Proceedings of Eighth International Joint Conference on Artificial Intelligence*, pages 951-1099. IJCAI83, August, 1983.

[6] Huertas, A., Cole, W. and Nevatia, R.
Detecting Runways in Aerial Images.
In *Proceedings of Sixth National Conference on Artificial Intelligence*, pages 712-717. AAAI, July, 1987.

[7] Kestner, Dr.-Ing. W. and Prof. Dr.-Ing. H. Kazmierczak.
*Semiautomatic Extraction of Roads From Aerial Photographs.*
Final Technical Report, Research Institute for Information Processing and Pattern Recognition, FIM, Karlsruhe, West Germany, June, 1978.

[8] McKeown, D.M.
MAPS: The Organization of a Spatial Database System Using Imagery, Terrain, and Map Data.
In *Proceedings: DARPA Image Understanding Workshop*, pages 105-127. June, 1983.
Also available as Technical Report CMU-CS-83-136.

| Image | Width | Length | Time | | Subjective Reason for Stopping |
|---|---|---|---|---|---|
| dc1523 | 10 [32] | 186 [596] | 4+22 | A | Curve / intersection |
| dc1523 | 6 [19] | 171 [548] | 4+30 | A | Poor edge strength |
| dc1418 | 5 [16] | 220 [702] | 5+31 | M | Road splits |
| dc38006 | 5 [5] | 385 [408] | 9+46 | A | Poor edge strength |
| dc1420 | 8 [26] | 188 [608] | 3+26 | A | Road splits |
| dc1419 | 5 [16] | 31 [97] | 1+7 | A | Poor edge strength |
| dc39206 | 10 [10] | 182 [183] | 5+25 | M | Found edges not on road |
| dc39206 | 8 [8] | 385 [388] | 6+45 | A | Anomaly |
| dc1522 | 15 [48] | 1066 [3432] | 21+12 | A | Cloverleaf ramps |
| dc1012 | 5 [25] | 212 [1067] | 3+28 | M | Shoulder widens |
| belv1 | 8 [8] | 36 [36] | 1+8 | A | Wrong edge followed |
| belv4 | 6 [6] | 20 [20] | 0+5 | A | Poor edges |
| belv4 | 6 [6] | 132 [132] | 3+19 | A | Poor edge strength |
| dc38006 | 8 [8] | 36 [38] | 1+8 | A | Poor edge strength |
| dc1421 | 6 [19] | 111 [355] | 2+15 | A | Intersection / poor edges |
| dc38617 | 8 [9] | 115 [126] | 2+12 | A | Edge of image |
| dc36809 | 12 [13] | 82 [88] | 1+12 | A | Curve in road / vehicles |
| dc1419 | 6 [19] | 184 [576] | 3+23 | M | Wrong edge followed / ramp |
| dc38008 | 12 [13] | 464 [488] | 8+56 | M | Road splits |
| dc1013 | 9 [46] | 193 [998] | 3+26 | A | Surface material change |
| dc38614 | 15 [17] | 26 [29] | 1+6 | A | Vehicles |
| dc1422 | 8 [25] | 46 [143] | 1+10 | A | Poor edges |
| dc36810 | 9 [10] | 205 [217] | 4+28 | M | Wrong edges followed |
| dc38617 | 7 [8] | 158 [173] | 2+21 | A | Poor edge strength / vehicles |
| dc38617 | 14 [15] | 70 [77] | 1+12 | A | Occlusions / vehicles |
| dc1014 | 6 [30] | 20 [99] | 1+5 | A | Poor edges |
| dc38006 | 7 [7] | 128 [136] | 2+18 | A | Vehicle |
| dc38617 | 8 [9] | 43 [47] | 1+8 | A | Unknown |
| dc37401 | 35 [36] | 1004 [1058] | 19+114 | A | Edge of image |
| dc1522 | 14 [45] | 35 [112] | 1+7 | A | Poor edge strength |
| dc38011 | 7 [7] | 49 [52] | 1+8 | A | Vehicle / occlusion / poor edge |
| dc1011 | 5 [26] | 20 [105] | 0+5 | A | Poor edges |
| dc1418 | 12 [38] | 22 [70] | 0+5 | A | Poor edges |
| dc37406 | 6 [6] | 306 [313] | 4+36 | A | Occlusion |
| dc38013 | 11 [11] | 58 [60] | 1+12 | A | Poor edges |

**Figure 7-1:** Using Edge Tracking As A Single Method (Training)


[9] McKeown, D.M.
The Role of Artificial Intelligence in the Integration of Remotely Sensed Data with Geographic Information Systems.
*IEEE Transactions on Geoscience and Remote Sensing* GE-25(3):330-348, May, 1987.
Also available as Technical Report CMU-CS-86-174.

[10] R. Nevatia and K. E. Price.
Locating Structures in Aerial Images.
*IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. PAMI-4(5):pp. 476-484, September, 1982.

[11] R. Nevatia and K. R. Babu.
Linear Feature Extraction and Description.
*Computer Graphics and Image Processing* vol. 13:pp. 257-269, July, 1980.

[12] Quam, Lynn H.
Road Tracking and Anomaly Detection in Aerial Imagery.
*Proceedings: Image Understanding Workshop* :pp. 51-55, May, 1978.

[13] Snedecor, George W. and William G. Cochran.
*Statistical Methods, Sixth Edition.*
The Iowa State University Press, Ames, Iowa, U.S.A., 1967.

[14] Vocar, J. M. and R. O. Faiss.
*Image Magnification: Elementary With Staran.*
Goodyear Aerospace Report GER-16342, Goodyear Aerospace, August, 1976.

| Image | Width | Length | Time | | Subjective Reason for Stopping |
|---|---|---|---|---|---|
| dc1322 | 8 [24] | 32 [101] | 1+7 | A | Poor edges |
| dc1417 | 10 [31] | 190 [596] | 3+31 | A | Poor edges |
| dc1417 | 5 [16] | 20 [62] | 1+5 | A | Poor edges |
| dc1417 | 5 [16] | 50 [156] | 2+11 | A | Intersection / poor edges |
| dc1419 | 5 [16] | 20 [62] | 1+5 | A | Poor edges |
| dc1419 | 7 [22] | 26 [81] | 1+6 | A | Poor edges |
| dc1419 | 15 [47] | 30 [93] | 1+6 | A | Surface change / poor edges |
| dc1420 | 8 [26] | 20 [64] | 1+5 | A | Poor edges |
| dc1420 | 6 [19] | 20 [64] | 1+5 | A | Fuzzy image |
| dc1421 | 6 [19] | 22 [70] | 0+5 | A | Poor edges |
| dc1422 | 8 [25] | 58 [180] | 1+9 | A | Poor edges |
| dc1522 | 7 [22] | 204 [657] | 4+27 | A | Width change |
| dc1522 | 7 [22] | 170 [547] | 3+26 | A | Road curves / poor edges |
| dc1522 | 6 [19] | 57 [183] | 1+9 | A | Intersection / road curves |
| dc1523 | 7 [22] | 46 [147] | 2+10 | M | Intersection / wrong edges followed |
| dc1524 | 7 [22] | 20 [63] | 1+5 | A | Unknown |
| dc1524 | 6 [19] | 20 [63] | 1+5 | A | Poor edges |
| dc1625 | 7 [22] | 23 [72] | 1+6 | A | Poor edges |
| dc1625 | 6 [19] | 20 [63] | 1+5 | A | Poor edges |
| dc1625 | 6 [19] | 20 [63] | 0+5 | A | Poor edges |
| dc1625 | 7 [22] | 24 [76] | 1+6 | A | Poor edges |
| dc36808 | 8 [9] | 184 [195] | 3+24 | A | Vehicle |
| dc36808 | 7 [7] | 34 [36] | 1+7 | A | Edges not constant width |
| dc36810 | 14 [15] | 148 [156] | 4+20 | A | Poor edges / road turns |
| dc36810 | 13 [14] | 44 [46] | 1+8 | A | Poor edges |
| dc36810 | 13 [14] | 24 [25] | 1+6 | A | Poor edges |
| dc36810 | 11 [12] | 20 [21] | 1+5 | A | Poor edges |
| dc38012 | 7 [7] | 35 [36] | 1+9 | A | Road turns |
| dc38612 | 10 [11] | 186 [195] | 4+30 | A | Parked vehicles |
| dc39203 | 8 [8] | 644 [668] | 13+86 | A | Vehicles |
| dc39204 | 8 [8] | 112 [111] | 2+16 | A | Intersection |
| dc39206 | 8 [8] | 51 [51] | 1+8 | A | Poor edges |
| dc39208 | 11 [11] | 185 [184] | 4+22 | M | Wrong edge followed |
| dc39208 | 15 [15] | 117 [116] | 2+16 | A | Turn too sharp |
| dc39208 | 8 [8] | 49 [48] | 2+12 | A | Poor edges |

Figure 7-2: Using Edge Tracking As A Single Method (Test)

| Image | Width | Length | Time | | Subjective Reason for Stopping |
|---|---|---|---|---|---|
| dc1523 | 10 [32] | 337 [1080] | 6+36 | M | Guess ahead found other lane |
| dc1523 | 6 [19] | 1539 [4976] | 28+198 | A | Edge of image |
| dc1418 | 5 [16] | 413 [1329] | 7+38 | M | Unknown |
| dc38006 | 5 [5] | 2025 [2145] | 41+148 | M | Image fuzzy at edge |
| dc1420 | 8 [26] | 315 [1019] | 8+65 | M | Surface change / poor contrast |
| dc1419 | 5 [16] | 317 [996] | 6+28 | M | Poor surface pattern / turn |
| dc39206 | 10 [10] | 2128 [2221] | 73+287 | M | Surface change / vehicles |
| dc39206 | 8 [8] | 2045 [2131] | 62+238 | A | Edge of image |
| dc1522 | 15 [48] | 1303 [4170] | 33+205 | A | Edge of image |
| dc1012 | 5 [25] | 65 [326] | 3+17 | M | Poor surface pattern |
| belv1 | 8 [8] | 432 [432] | 11+51 | A | Edge of image |
| belv4 | 6 [6] | 161 [161] | 5+29 | A | Poor surface pattern |
| belv4 | 6 [6] | 121 [121] | 6+34 | A | Poor surface pattern |
| dc38006 | 8 [8] | 166 [667] | 5+31 | A | Overpass |
| dc1421 | 6 [19] | 117 [1731] | 3+17 | M | Poor contrast: road/background |
| dc38617 | 8 [9] | 113 [687] | 6+37 | A | Edge of image |
| dc36809 | 12 [13] | 624 [5026] | 18+98 | A | Edge of image |
| dc1419 | 6 [19] | 548 [168] | 11+56 | M | Poor surface pattern |
| dc38008 | 12 [13] | 652 [290] | 37+107 | M | Road splits |
| dc1013 | 9 [46] | 954 [1223] | 40+136 | M | Overpass |
| dc38614 | 15 [17] | 153 [313] | 15+38 | M | Intersection |
| dc1422 | 8 [25] | 93 [286] | 4+20 | M | Shadow / S-turn |
| dc36810 | 9 [10] | 1157 [1223] | 59+189 | A | Intersection / surface change |
| dc38617 | 7 [8] | 288 [313] | 13+63 | A | Overpass |
| dc38617 | 14 [15] | 262 [286] | 27+104 | M | Intersection |
| dc1014 | 6 [30] | 72 [356] | 5+29 | A | Poor surface pattern |
| dc38006 | 7 [7] | 32 [34] | 3+9 | M | Curvature too great |
| dc38617 | 8 [9] | 150 [164] | 5+24 | A | Overpass |
| dc37401 | 35 [36 | 1006 [1060] | 112+482 | A | Edge of image |
| dc1522 | 14 [45] | 556 [1798] | 34+108 | M | End of road |
| dc38011 | 7 [7] | 126 [135] | 8+32 | M | Poor surface pattern |
| dc1011 | 5 [26] | 1581 [8312] | 29+119 | M | Road splits |
| dc1418 | 12 [38] | 1066 [3424] | 26+139 | M | Road splits |
| dc37406 | 6 [6] | 1277 [1308] | 33+129 | M | Shadows from trees |
| dc38013 | 11 [11] | 1642 [1744] | 38+193 | A | Edge of image |

Figure 7-3: Using Surface Tracking As A Single Method (Training)

| Image | Width | Length | Time | | Subjective Reason for Stopping |
|---|---|---|---|---|---|
| dc1322 | 8 [24] | 197 [624] | 5+30 | M | Poor surface pattern |
| dc1417 | 10 [31] | 1466 [4734] | 34+176 | M | Background material same road |
| dc1417 | 5 [16] | 294 [926] | 8+36 | M | Intersection |
| dc1417 | 5 [16] | 93 [292] | 2+11 | M | Poor contrast in image |
| dc1419 | 5 [16] | 62 [194] | 2+12 | A | Poor contrast in image |
| dc1419 | 7 [22] | 873 [2778] | 16+78 | A | Road splits |
| dc1419 | 15 [47] | 490 [1545] | 18+110 | M | Surface markings (runway) |
| dc1420 | 8 [26] | 243 [786] | 8+50 | M | Poor surface pattern |
| dc1420 | 6 [19] | 57 [184] | 3+15 | M | Image fuzzy |
| dc1421 | 6 [19] | 405 [1294] | 8+42 | M | Shadow |
| dc1422 | 8 [25] | 265 [828] | 5+38 | M | Background material same road |
| dc1522 | 7 [22] | 537 [1737] | 17+66 | M | Background material same road |
| dc1522 | 7 [22] | 135 [435] | 3+19 | A | Overpass |
| dc1522 | 6 [19] | 165 [532] | 6+23 | M | Poor surface pattern / shadows |
| dc1523 | 7 [22] | 117 [375] | 4+20 | M | Poor surface pattern |
| dc1524 | 7 [22] | 769 [2458] | 13+75 | M | Overpass |
| dc1524 | 6 [19] | 131 [416] | 3+18 | M | Background material same road |
| dc1625 | 7 [22] | 49 [155] | 2+14 | A | Low contrast / edge width vary |
| dc1625 | 6 [19] | 115 [365] | 2+13 | M | Road widens |
| dc1625 | 6 [19] | 934 [2984] | 13+76 | M | Edge of image |
| dc1625 | 7 [22] | 169 [537] | 2+17 | M | Poor surface pattern |
| dc36808 | 8 [9] | 741 [789] | 12+68 | A | Edge of image |
| dc36808 | 7 [7] | 165 [176] | 2+18 | M | Intersection / width change |
| dc36810 | 14 [15] | 1107 [1170] | 41+216 | M | Low contrast on road |
| dc36810 | 13 [14] | 348 [368] | 19+76 | M | Poor surface pattern / turn |
| dc36810 | 13 [14] | 285 [301] | 16+58 | A | Poor surface pattern / turn |
| dc36810 | 11 [12] | 1735 [1833] | 57+263 | A | Overpass |
| dc38012 | 7 [7] | 74 [77] | 2+15 | M | Width Change |
| dc38612 | 10 [11] | 263 [275] | 5+40 | M | Low contrast on road |
| dc39203 | 8 [8] | 2227 [2310] | 70+271 | A | Vehicles / intersection |
| dc39204 | 8 [8] | 410 [414] | 18+63 | M | Poor surface pattern |
| dc39206 | 8 [8] | 344 [346] | 16+59 | M | Road widens |
| dc39208 | 11 [11] | 1289 [1321] | 51+183 | M | Intersection |
| dc39208 | 15 [15] | 108 [107] | 5+23 | M | Road turns |
| dc39208 | 8 [8] | 1174 [1200] | 28+126 | A | Overpass |

Figure 7-4: Using Surface Tracking As A Single Method (Test)

| Image | Width | Length | Time | | Subjective Reason for Stopping |
|---|---|---|---|---|---|
| dc1523 | 10 [32] | 582 [1868] | 46+348 | M | Road widens |
| dc1523 | 6 [19] | 1497 [4838] | 93+725 | M | Road splits |
| dc1418 | 5 [16] | 841 [2696] | 46+288 | M | Poor edges / low contrast |
| dc38006 | 5 [5] | 2140 [2267] | 104+621 | M | Image fuzzy at edge |
| dc1420 | 8 [26] | 228 [737] | 26+102 | M | Road splits |
| dc1419 | 5 [16] | 351 [1103] | 48+328 | M | Road turns and widens |
| dc39206 | 10 [10] | 2130 [2223] | 142+770 | M | Surface change / width change |
| dc39206 | 8 [8] | 2037 [2122] | 138+826 | A | Edge of image |
| dc1522 | 15 [48] | 1300 [4161] | 81+464 | A | Edge of image |
| dc1012 | 5 [25] | 212 [1067] | 16+82 | M | Road shoulder widens |
| belv1 | 8 [8] | 423 [423] | 53+232 | A | Edge of image |
| belv4 | 6 [6] | 83 [83] | 26+158 | M | Poor edges / surface pattern |
| belv4 | 6 [6] | 132 [132] | 11+63 | A | Poor edges / surface pattern |
| dc38006 | 8 [8] | 166 [176] | 11+103 | A | Overpass |
| dc1421 | 6 [19] | 177 [566] | 11+80 | M | Intersection |
| dc38617 | 8 [9] | 115 [126] | 7+50 | A | Edge of image |
| dc36809 | 12 [13] | 624 [667] | 36+277 | A | Edge of image |
| dc1419 | 6 [19] | 1076 [3440] | 64+495 | M | Overpass |
| dc38008 | 12 [13] | 514 [541] | 27+149 | M | Road splits |
| dc1013 | 9 [46] | 953 [5020] | 53+409 | M | Overpass / turn |
| dc38614 | 15 [17] | 186 [204] | 38+254 | M | Intersection / parked vehicles |
| dc1422 | 8 [25] | 107 [334] | 12+76 | M | Shadow from trees / turn |
| dc36810 | 9 [10] | 749 [791] | 55+395 | M | Intersection |
| dc38617 | 7 [8] | 288 [314] | 20+158 | A | Overpass |
| dc38617 | 14 [15] | 262 [286] | 36+213 | M | Intersection / vehicles |
| dc1014 | 6 [30] | 90 [446] | 29+200 | M | Poor contrast in image |
| dc38006 | 7 [7] | 58 [61] | 10+42 | M | Turn too sharp |
| dc38617 | 8 [9] | 150 [164] | 8+59 | A | Overpass |
| dc37401 | 35 [36] | 1004 [1058] | 106+672 | A | Edge of image |
| dc1522 | 14 [45] | 567 [1834] | 60+422 | A | End of road (runway) |
| dc38011 | 7 [7] | 128 [137] | 14+87 | M | Surface pattern / width change |
| dc1011 | 5 [26] | 67 [352] | 8+82 | M | Poor edges |
| dc1418 | 12 [38] | 1075 [3454] | 123+730 | M | Road splits |
| dc37406 | 6 [6] | 1254 [1284] | 128+432 | M | Shadow on road |
| dc38013 | 11 [11] | 1642 [1743] | 100+857 | A | Edge of image |

Figure 7-5: Using Combined Edge And Surface Tracking (Training)

| Image | Width | Length | Time | | Subjective Reason for Stopping |
|---|---|---|---|---|---|
| dc1322 | 8 [24] | 460 [1457] | 48+388 | A | Poor contrast in image |
| dc1417 | 10 [31] | 1501 [4851] | 113+925 | M | Background same as road |
| dc1417 | 5 [16] | 298 [938] | 25+233 | M | Intersection |
| dc1417 | 5 [16] | 121 [380] | 9+79 | M | Poor surface pattern/contrast |
| dc1419 | 5 [16] | 62 [194] | 6+44 | A | Poor surface pattern/intersection |
| dc1419 | 7 [22] | 865 [2752] | 61+590 | M | Road splits |
| dc1419 | 15 [47] | 494 [1558] | 44+420 | M | Surface markings (runway) |
| dc1420 | 8 [26] | 281 [909] | 41+289 | M | Poor surface pattern/varied width |
| dc1420 | 6 [19] | 57 [184] | 7+51 | M | Poor contrast in image |
| dc1421 | 6 [19] | 417 [1332] | 42+308 | M | Shadow |
| dc1422 | 8 [25] | 259 [809] | 24+202 | M | Median widens |
| dc1522 | 7 [22] | 688 [2225] | 48+403 | A | Road turns/poor surface pattern |
| dc1522 | 7 [22] | 591 [1912] | 44+308 | A | Road turns/poor surface pattern |
| dc1522 | 6 [19] | 188 [606] | 21+178 | M | Intersection |
| dc1523 | 7 [22] | 128 [410] | 17+143 | M | Road widens/poor surface pattern |
| dc1524 | 7 [22] | 784 [2507] | 81+573 | M | Overpass |
| dc1524 | 6 [19] | 61 [194] | 5+45 | M | Intersection/poor surface pattern |
| dc1625 | 7 [22] | 49 [155] | 5+40 | A | Road widens |
| dc1625 | 6 [19] | 105 [333] | 12+86 | M | Road widens |
| dc1625 | 6 [19] | 948 [3029] | 98+705 | M | Edge of image |
| dc1625 | 7 [22] | 128 [406] | 10+99 | M | Poor surface pattern |
| dc36808 | 8 [9] | 725 [772] | 42+267 | A | Edge of image |
| dc36808 | 7 [7] | 200 [213] | 12+98 | M | Intersection |
| dc36810 | 14 [15] | 1596 [1686] | 129+855 | A | Edge of image |
| dc36810 | 13 [14] | 369 [390] | 35+260 | A | Poor surface pattern |
| dc36810 | 13 [14] | 374 [395] | 51+366 | M | Poor surface pattern |
| dc36810 | 11 [12] | 1527 [1613] | 102+730 | M | Intersection/wrong edge followed |
| dc38012 | 7 [7] | 88 [92] | 10+83 | M | Road widens |
| dc38612 | 10 [11] | 1608 [1613] | 94+633 | M | Road splits |
| dc39203 | 8 [8] | 2232 [2316] | 142+856 | A | Vehicles / intersection |
| dc39204 | 8 [8] | 416 [420] | 36+293 | M | Poor surface pattern / turn |
| dc39206 | 8 [8] | 553 [559] | 44+354 | M | Road same as background material |
| dc39208 | 11 [11] | 1314 [1348] | 99+610 | M | Intersection |
| dc39208 | 15 [15] | 134 [133] | 11+72 | A | Road turns/poor surface pattern |
| dc39208 | 8 [8] | 416 [417] | 36+293 | M | Poor surface pattern / turn |

**Figure 7-6:** Using Combined Edge Surface Tracking (Test)

| > | E | S | C |
|---|---|---|---|
| E | - | 3 | 1 |
| S | 30 | - | 6 |
| C | 30 | 12 | - |

Tuned Dataset

| > | E | S | C |
|---|---|---|---|
| E | - | 1 | 0 |
| S | 33 | - | 5 |
| C | 35 | 13 | - |

Test Dataset

| = | E | S | C |
|---|---|---|---|
| E | - | 2 | 4 |
| S | 2 | - | 17 |
| C | 4 | 17 | - |

Tuned Dataset

| = | E | S | C |
|---|---|---|---|
| E | - | 1 | 0 |
| S | 1 | - | 17 |
| C | 0 | 17 | - |

Test Dataset

**Figure 7-7:** Performance Comparison Of Tracker Combinations

# -PACE-
# An Evironment For Intelligence Analysis [1]

N.R Corby, J.L, Mundy and P.A. Vrobel

General Electric Corporate Research and Development Center
Schenectady, NY 12301

A.J. Hanson, L.H. Quam, G.B. Smith and T.M. Strat

Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

**Abstract**

The system design for an environment to aid intelligence analysis is described. The goal of this system development is to provide an integrated set of tools for analysing the status of a military site from a set of aerial images and collateral reports. The system, called PACE, is to be implemented by integrating various research tools that have been developed at SRI and GE. The system development will be focussed on the application of airbase monitoring. Experiments are underway at the 109th Tactical Airlift Group Base in Schenectady.

## 1 Introduction

### 1.1 The PACE Concept

The Perceptual Analysis and Control Environment, or PACE, system is currently under development at GE in cooperation with SRI. The motivation for the development of this system is to augment the existing procedures for carrying out intelligence analysis from aerial photographs and collateral reports. The main function of the PACE system is to compile images from multiple image sources into a common three dimensional reference frame and permit the visualization of this data through synthetic image display, including image projection mapping.

PACE is also intended to permit the integration of various geometric constraint information available from sources other than images, such as functional descriptions and other intelligence reports. The ultimate goal for PACE is to be able to assist in assessing the integrity of the existing site description and to aid in the planning of additional image acquisition tasks.

The typical intelligence analysis task is motivated by a set of queries which are to be satisfied by examining a set of aerial images of the site. These images are acquired by reconnaissance aircraft or satellites which are deployed in response to the queries. An important aspect is making optimal use of these resources so that the intelligence analysis can be carried out expediently and at reasonable cost. For example, there may be a number of outstanding queries that can be satisfied during a given orbital pass of a satellite.The repositioning of the satellite requires fuel, so it may be desirable to queue up a number of image requests for a given satellite alignment.

Previous reports and associated imagery are often available for the site as well as other sources, such as eyewitness accounts and message traffic. The key benefit of the PACE system is the coordination of these information elements into a common environment using image understanding tools to assist the analyst in object recognition and site status evaluation.

In the development of the initial PACE prototype we plan to focus on three primary sources of site intelligence:

- visual aerial photographs
- infra-red aerial photographs
- natural language reports

These sources are used to derive a description of the site based on a detailed a priori model of the site and models of the expected site elements.

### 1.2 An Application Scenario

To illustrate the intended PACE functions, consider the problem of airport monitoring. We assume that the monitoring is carried out on a routine basis and that a detailed representation of the buildings, aircraft and vehicles are available.

The functions of the monitoring process are:

- Determine the existence of new buildings and alterations in the terrain.
- Determine the classification and location of vehicles and aircraft.
- Determine the nature of current missions underway at the base.

The full automation of these tasks is beyond the grasp of current image understanding techniques. However, there are some emerging techniques in model-based object recognition and geometric constraint analysis that can assist the analyst and carry out the more routine, straightforward aspects of these airport monitoring tasks. A central goal of PACE is the integration of these techniques into a uniform environment and with a convenient user interface to permit a close interaction between the analyst and the IU tools.

[1] This work was supported in part by the DARPA Strategic Computing Vision Program in conjunction with the Army Engineer Topographic Laboratories under Contract No. DACA76-86-C-0007.

### 1.3 Requirements For PACE

**Sensor Fusion** It is seldom the case that images are registered in a known coordinate reference frame. In order to compare a set of images from the site it is necessary to map the image data into a common reference frame. This compilation process is commonly referred to as *sensor fusion*. The fusion process requires that common reference points are located in the images and then the transformation between the images is computed from this set of common references.

The current approaches generally involve manual selection of the common features, and then an optimization process to provide an accurate estimate for the transformation between the coordinate frames of the cameras that generate each image [Photogr.Man.]. It is desirable that these correspondences be determined more automatically, perhaps interactively guided by a segmentation of the image data.

An important application of this image fusion process for the intelligence analysis is the ability to assess the coverage of the site by the available images. To make this assessment more convenient, the image data is mapped onto the surfaces of the object models according to the transformations derived from the fusion process. The analyst can then generate arbitrary views of the site to review the best available resolution and coverage for features of interest.

**Collateral Constraints** In addition to information taken directly from images, it is generally the case that collateral information about the site is available in reports. These reports refer to qualitative relationships concerning the objects at the site, e.g. "the fuel truck is parked near the hanger." It is a goal of PACE to be able to use these types of constraints to aid in the object recognition and fusion process.

It is also the case that geometric relationships that are derived from imagery may be useful in determining the functional status of vehicles at the site. For example, if a fuel truck is near an aircraft, it is reasonable to conclude that it is in the process of refueling the aircraft.

**Resource Allocation** Another requirement that must be satisfied by the PACE system is the optimization of sensor coverage. The analyst should be able to review the supporting evidence for a particular object classification or geometric relation and be able to decide if more images or images of higher resolution are required to reach an adequate level of confidence in the conclusion.

The system will support both visual and infra red image collection and the resource allocation must take into account the specific constraints that can be extracted from each image modality.

**The User Interface** The user interface must support many demanding requirements in the intelligence analysis application. We have already mentioned the use of image mapping onto object surfaces as a means of displaying the fused sensor data. The user interface must also support these additional functions:

- The interactive generation of object models.
- Animation of viewpoint trajectories.
- Interactive object classification and pose constraint.

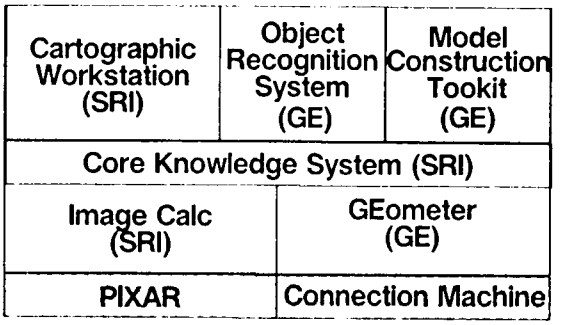



Figure 1: The PACE prototype architecture

- Intelligence report generation.

In the prototype system under development, we are only beginning to scratch the surface of these requirements. However, the PACE system concept has proven to be a good focus for our ongoing research into geometric reasoning and object recognition.

## 2 The PACE Prototype

The intial PACE architecture is shown in Figure 1 The figure shows how the various systems under development at SRI and GE will be integrated, as well as the hardware support for carrying out the required computation. These systems have been developed independently but under a common environment Symbolics ZETA-LISP. Both SRI and GE use Symbolics Lisp Machines with 8 or 24 bit color display hardware. In addition, GE has been using SRI's Image-Calc$^{TM}$ system, as a basic image processing environment, since 1984. These commonalities will make the integration easier than would ordinarily be the case.

We now proceed to describe in more detail the functionality of each of the components.

### 2.1 The Cartographic Workstation

The Cartographic Workstation [Hanson et al], is directed at the problem of creating a world description from a number of sources of information. The world description consists of polyhedral object and terrain models as well as three dimensional boundary spline curves. This description is generated from the following sources:

- Images
- Digital Terrain Elevation Data
- Compiled Feature Data
- Feature Models

In this context, feature refers to such objects as buildings, roads or other structures which need to be identifed individually on maps.

A key aspect of the Cartographic Workstation is the user interface which supports a wide variety of interactive modes for manipulating geometric and image objects. The system can support a number of coordinated viewpoints at the same time, so the user can observe the three dimensional configuration of

the database. The mouse interface is designed to provide a smooth access to image data, three dimensional spline curves and polyhedral object models.

Another important function is the rendering of object surfaces by conventional synthetic light models or more importantly, by image mapping onto the object surfaces. This mapping is carried out by attaching image intensity array data to each polyhedral face of the object and terrain models. When a view of the world data base is created, the intensity maps are used to generate a realistic presentation of the object surfaces.

A primary function of the user interface is the creation of feature models by interacting with multiple image views and with digital terrain elevation data. In many cases, the elevation data is generated from stereo image pairs.

The object models are parameterized, and the model parameters can be manipulated through mouse interaction. The user adjusts the model parameters to give an acceptable fit for the model in the available image views. Some functions are also under development to use image segmentation to assist the delineation of boundary curves.

The Cartographic Workstation also maintains a hierarchy of coordinate transformations and a data base structure for representing the relationships between objects in the world. A significant portion of the effort needed to create the data base is involved in defining these transformations and insuring that the objects are correctly placed with respect to each other. Tools are available for computing transformations from various constraints, such as a set of common image feature locations.

In a practical cartographic system, there are many forms of coordinate representation that must be related. These special coordinate systems are nicely represented in the Symbolics object-oriented *flavor* system. For example, the general operation of the perspective transformation forms the base flavor with various *mixins* to handle increasingly specific coordinate systems.

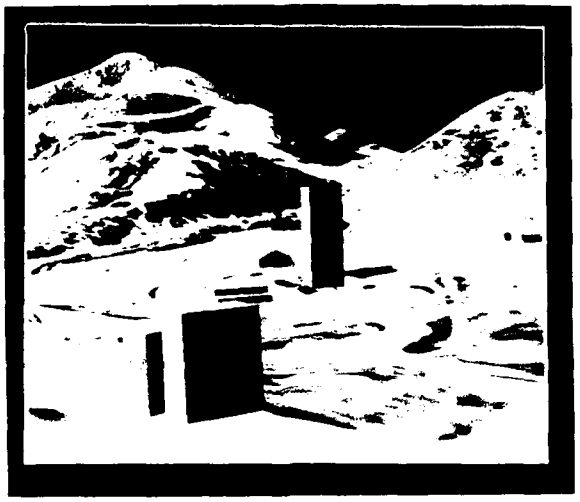


Figure 2: A rendered view of a three dimensional data base created by the Cartographic Workstation.

An example of the result of a database generation exercise is shown in Figure 2

## 2.2 GEometer

A system for representing and reasoning about geometry, under development at GE, is shown in Figure 3. The GEometer$^{TM}$ structure represents the geometric and topological relationships that exist in the world as well as mechanisms for representing the process of projecting sensor viewpoints into two dimensional image and segmentation data. The general organization is in terms of the concept of inclusion or composition.

Each object has its own coordinate reference frame and the world data structure maintains the relationship between these coordinate systems. Closely linked to the world representation, is a set of sensors and associated viewing transformations that link the coordinates system of the sensor image planes to the world coordinate system. Each sensor is described in terms of the associated geometry of projection and the image signal processing model.

**Object Topology** The object structure is shown in Figure 4. First we describe the topological structure of an object, which follows the standard formal convention [Wesley and Markowsky]. The object is composed of *blocks* which are closed volumes bounded by a cycle of surface faces. This closed chain of faces is called a *2-cycle*. A block can represent either material or empty space, so that full Constructive Solid Geometry (CSG), object descriptions can be created.

The *face* is a closed region of surface which is bounded by a closed sequence of edges, called a 1-cycle. The surface itself is a plane, for the case of a polyhedral object.

The *edge* is a bounded one dimensional curve which corresponds to a line segment for the polyhedral case. The edge is

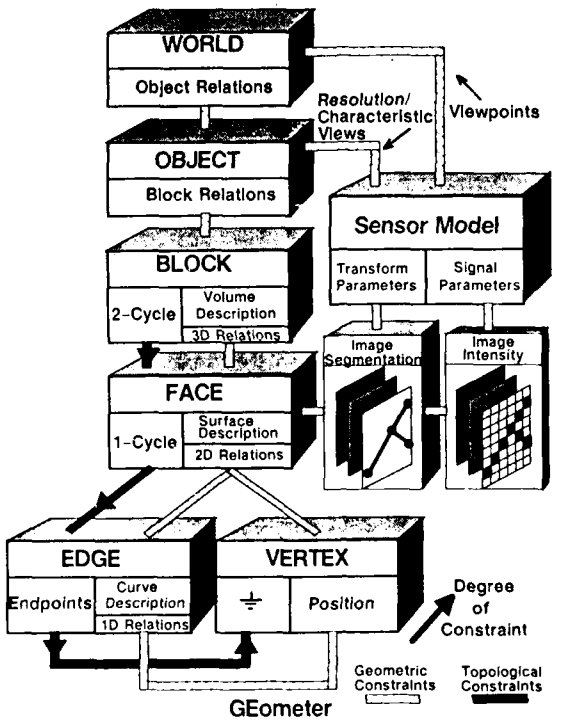



Figure 3: The GEometer representation. The structure accounts for multiple sensor views of the three dimensional world.

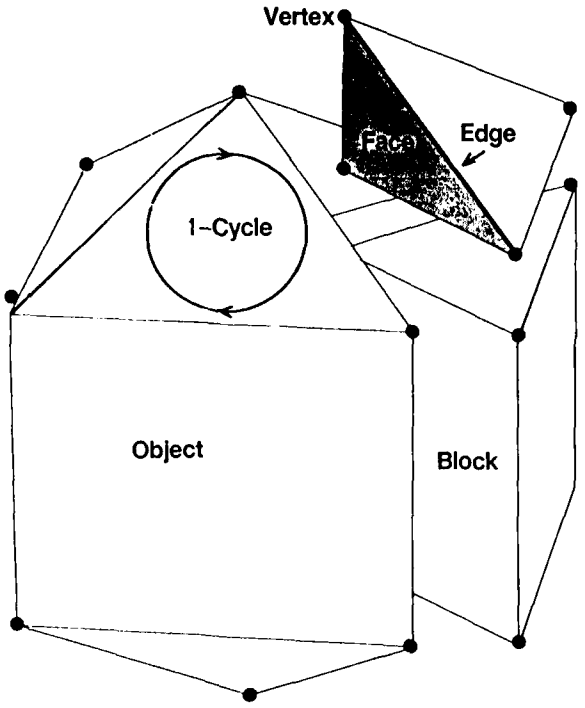



Figure 4: The topological structure of an object.

described by a parametric curve description which defines the points lying on the edge. The edge is bounded by two endpoints which are each defined by a vertex.

The *vertex* is considered to be zero dimensional and defines a point in space.

**Object Geometric Operations and Relations** In addition to the topological structure just introduced, there is a set of fundamental relationships and operations that are defined for the object components. The relationships and operations can be classified according to the dimension of the space that is needed to contain the object components involved in the relationship. For example, the ordering of points on a line, or the length of a line segment are considered to be one dimensional, or scalar operations.

Computing the intersection of a set of coplanar line segments or testing the collinearity of a set of points are considered to be two dimensional procedures; the coplanarity of a set of points or testing the intersection of two faces are classified as three dimensional.

The complete set of operations supported by GEometer are generated from the fundamental geometric predicates defined by Hilbert. These four predicates are: *equality,intersect,between*, and *equidistant*. Hilbert demonstrated that all other geometric operations and relations can be generated from these primitives. For example, the predicate *between(a b c)* tests whether point b is between points a and c on a line. This relation can be used to define the concept of a point being one side or the other of a given line. This sidedness relation can, in turn, determine whether or not a point is *inside or outside of a boundary*. This containment test is the main computation involved in boolean intersection and display processing.

**Generic Geometric Relations and Operations** Perhaps the most significant contribution of the proposed representation is the ability to express symbolic geometric constraints [Mundy]. The representation as described so far is similar to many solid modeling systems; but perhaps more formally defined. However, such modeling systems, (e.g. PADL [Brown]), manipulate fixed numerical descriptions of object vertex coordinates and surface parameters. All operations and relations are carried out using arithmetical computation.

In the proposed representation, the geometric and topological constraints can be expressed more generically in a lattice structure, as illustrated in Figure 5. The lattice represents increasing degrees of constraint, as indicated by the arrow relations. In the example, we start with the general concept of the polyhedron which can be constrained in many ways. The geometry of the polyhedron can be constrained to be convex, where a line segment with both endpoints inside the polyhedron must also lie entirely inside the polyhedron. There are many specializations of the convex polyhedron, including the regular polyhedron which has all faces the same, and the rectangular prism which is a six sided polyhedron with rectangular faces. The notion of a lattice structure is shown by the example that various subtypes can be specialized to a common type. The regular polyhedron and the rectangular prism can both be specialized to a cube. In the first case, the faces of the regular polyhedron are constrained to be squares. In the second case,

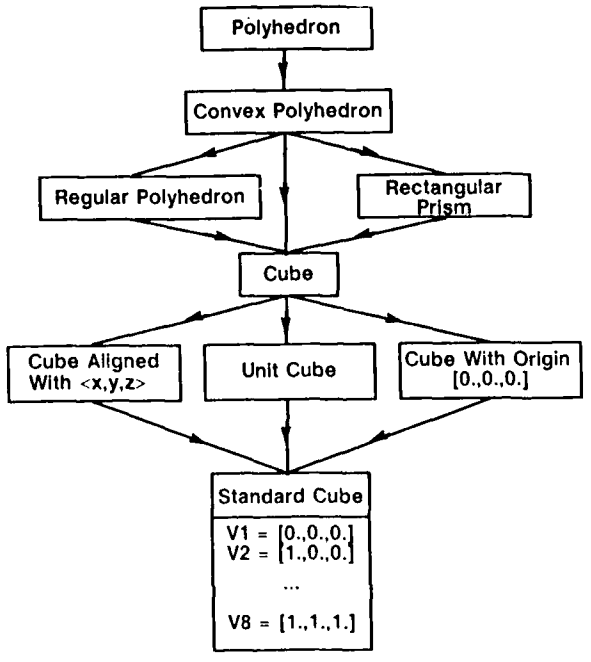



Figure 5: A lattice of constraint relations for polyhedral objects.

the lengths of the sides of the rectangular faces are made equal.

Note that so far there is nothing to constrain the actual size, location or orientation of the cube. Figure 5 shows how the cube can be further specified in terms of these constraints. Finally, the object becomes totally specified and the representation is then equivalent to a conventional solid polygonal model.

There are two major advantages obtained by extending the representation to include symbolic constraints:

- A wide variety of objects can be represented in terms of the parameters of the constraints. The constraints are expressed as equations and specifications that contain parameters that generate all objects satisfying the description. A specific object instance is represented by assigning values to all of the parameters.
- The symbolic constraints can be used to insure consistency. In current modeling systems, the numeric operations are not guaranteed to produce correct and consistent results. For example, the edges obtained by intersecting two faces may not lie in either face due to numerical roundoff error; however, it can be easily deduced that they must from the symbolic description of the process of face intersection and the description of the structure of the two faces.

**Sensor Models and Image Projection** The *sensor model* represents the geometric properties of image projection as well as the intensity signal processing models appropriate to a particular sensor mode. There can be a suite of sensor types along with a set of viewpoints for each sensor.

Referring back to Figure 3, each viewpoint generates an image as well as a corresponding geometric segmentation. The image is considered to be a planar surface and the segmentation elements lie in this plane. The segmentation is described in terms of a set of coplanar faces which are bounded by one-cycles of two dimensional edges and vertices. The edges and vertices form sets of connected chains. The segmentation elements are due to projections of three dimensional objects from a particular viewpoint. The edges may be due to geometric occlusion or shadowing, or produced by reflectance variations interior to an object face.

The location and orientation of the segmentation features in the image plane are geometrically constrained by the sensor projection model and the object description. Several examples should clarify the definition and application of this structure:

- Model Matching - The transformation between world, object and sensor coordinate frames are unknown, or partially specified. Assignments of segmentation features to model features provides further constraints on the transformations until consistent, unique transformations are determined.
- Stereo - Two, or more, sensor viewpoints are defined with a known transformation between the sensor reference frames. The correspondence of segmentation features between images constrains the three dimensional coordinates of the features.
- Calibration - The three dimensional coordinates of a world description are given, along with the corresponding image segmentation features. The correspondences are used to determine the sensor coordinate frame and sensor geometric parameters.

The idea of symbolic constraints is also relevant to such sensor relations. It is often the case that we know partial information about the sensor transformation which can reduce the computation of matching process. For example, constraints like: *Buildings are anchored to the ground plane.* or *The aircraft viewpoint is nearly vertical.* can be expressed as symbolic geometric relations.

The edge and region segmentation procedures operate on the image intensity data and produce geometric features which are described in the same format as the object *face* structure. This representation provides a uniform treatment across the system for all geometric entities.

The current status of the GEometer development is :

- A strictly numerical form of GEometer (previously called Geo-Calc) has been under development for several years and currently provides the basis for a model-based object recognition system [Thompson and Mundy] and for automated model generation tools [Connolly and Stenstrom].
- An separate experimental system for representing and manipulating symbolic geometric constraints (previously called GEometer) has also been implemented. This system has been used to prove a wide variety theorems in plane geometry and for consistency analysis in object viewing. This experimental system is currently being extended to handle inequalities which are needed to represent uncertain, empirically derived geometry.

The GEometer system is closely coupled to Image-Calc. It provides the user interface mechanism, and basic image manipulation capability. GEometer provides edge segmentation and [Canny] and boundary segmentation [Asada and Brady] algorithms within the Image-Calc enviroment. These segmenation algorithms provide a bridge between the image data representation and the GEometer geometric entity representation.

## 2.3 The Object Recognition System

GE has implemented a model-based object recognition system that determines the pose of a given object model in an intensity image. The model matching is based on a simple segmentation feature, the *vertex-pair* [Heller et al]. In this context, a vertex is a point defined by the intersection of two or more line segments in the boundary segmentation. The vertex-pair is formed by grouping two vertices and the two line segments associated with one of the vertices. It can be shown that the vertex-pair provides enough constraints to determine the affine transformation between the image coordinate system and the three dimensional model coordinate frame.

That is, a three dimensional vertex-pair is grouped in the object model and assigned a hypothesised corresponding vertex-pair in the segmentation. The model-to-image transformation is then computed for the assignment. The set of correct assignments is determined by clustering the transform values for each of the possible assignments of model vertex-pairs to segmentation vertex-pairs [Thompson and Mundy]. The matching process has a cost proportional to $MN^2$ where M is the number of model vertex-pairs and N is the number of vertices in the image segmentation.

Initial experiments have indicated that it is possible to obtain match transformations with rotational accuracy of about 5° and translational accuracy of several pixels. In the PACE system this intial estimate will be used to focus the search for additional reference features by constraints obtained from the initial match. As the number of matched features is increased, the accuracy of the transformation of a given image viewpoint transformation also increases [Lowe]. Conversely, a more accurate specification of the viewing transformation will allow incorrect feature assignments to be identified.

This object recognition system will also be exploited to verify the status of vehicles and other movable structures at the site. Part of the common routine analysis tasks involve counting and classification of familiar objects. In this application, each object model in the site library is matched against the scene and if a consistent set of feature assignments can be found in the segmentation, then the object and its location are determined.

## 2.4 Model Construction System

GE has also been developing a system to generate object models from a number of sources:

- Multiple Intensity Views
- Three Dimensional Range Data

Both of these systems are based on the concept of backprojecting the segmented image projection to form a solid which bounds the occluded space of the object view. These *viewsolids* are then combined across mulitple viewpoints by forming their

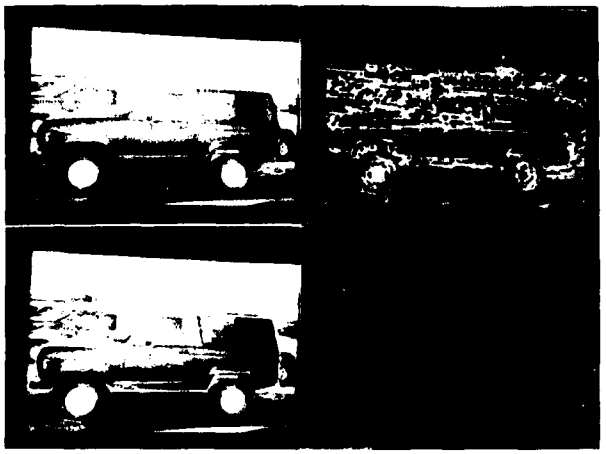


Figure 6: Segmentation guided vertex selection.

boolean intersection. The resulting intersection represents an outer bound of the object surface, consistent with all of the views [Connolly and Stenstrom]. The ultimate goal is the automatic generation of object models.

In addition, we have begun the development of some tools for using segmentation to guide the manual delineation of model features. One example is shown in Figure 6. The vertex locations are used to "snap" mouse driven interactive vertex selection during the construction of object faces in mulitple views. The upper left pane shows a view of the object. The upper right hand pane shows the image segmentation into edges and vertices. The lower left pane shows an isolated face of the object, together with a new face being extracted on the wheel of the vehicle. The lower right pane shows the three dimensional geometry of the isolated face by rotating the simulated camera viewpoint segmented into edges and vertices.

SRI is also developing segmentation guided boundary definition, where boundary splines are allowed to conform, in an error minimization process, with respect to intensity boundary constraints [Hanson et al].

## 2.5 CKS

The Core Knowledge System, or CKS, is under development at SRI [Smith and Strat]. The main functions of the system are to provide a mechanism for communication among various processes and to maintain a database of information about the world that is to be exchanged between processes.

There are two major data structures employed by CKS to represent the world, an octree for storing the location of objects, and a semantic network for describing relationships between objects. The octree provides a convenient way of efficiently computing spatial relationships. The semantic network provides a general way of storing functional and inheritance relationships.

The general control environment of CKS is a set of concurrent processes that exchange database queries and data tokens. This control is augmented by daemons which may be attached to any slot of any data token in the data base. The daemon can interrupt a specified process if the data in that slot is altered by another process.

This "community of processes" architecture adopted for the CKS requires that processes be able to communicate their opinions with one another and without undue interference from processes with competing views. This function is accomplished in CKS by representing database entries as *opinions* held by individual processes. The integration of these opinions relate to query language qualifiers such as, "APPARENTLY," which is valid if any process believes the predicate in the query. The qualifier, "POSSIBLY," holds if no process believes that the predicate is false and at least one agent believes that the fact is true.

A primary feature of the CKS is a capability for characterizing and retrieving information based upon the semantic content. In this regard, CKS must support a common vocabulary so that two processes wishing to communicate will find common data representations for the information to be exchanged. A second role of the semantic representation is to provide a basic set of relations between entities and concepts that proves useful for the majority of the processes.

A sample of a semantic data base represented in CKS is shown in Figure 7. The display was generated by the GRASPER semantic network display tool. GRASPER also provides a convenient interface for creating and updating the semantic database.

In the PACE application that was outlined above, CKS will provide the mechanism for representing the functional/spatial relationships that occur during the evolution of missions at an airbase. A great deal of specialized knowledge about the vehicles and mission functions must be available in order to interpret natural language reports and to integrate the relationships

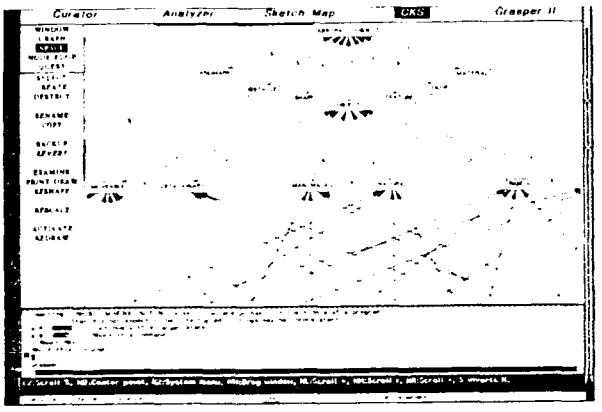



Figure 7: A semantic network in CKS.

Figure 8: A view of the 109th airfield.

derived from the reports with constraints and observations derived from geometric analysis.

It is also expected that CKS can provide information about an object which is useful in detection or classification. For example, it is very useful to know that certain vehicles are generally confined to a small number of specific areas.

## 3 The 109th Tactical Airlift Group

We have selected an Air National Guard Airbase near the GE Research and Development Center in Schenectady, New York as a source of realistic test images and mission requirements. The 109th Tactical Airlift Group maintains a fleet of C130 aircraft which carry out a wide variety of missions in diverse areas of the world. The C130 is a four engine, medium size transport airplane designed to carry personnel, vehicles and cargo. The C130 is able to operate from almost any kind of paved or unpaved airfield surface. For example, a recent mission involved the delivery of supplies to the US scientific base in Antarctica, using C130's with ski landing gear.

The base has agreed to help us in carrying out experiments to support the development of PACE. Information has already been provided about the details of airbase buildings and high resolution terrain elevations in the vicinity of the base. We have also gained permission to take aerial photographs of the base while missions are underway. It is fortuitous that the 109th carries out frequent evening missions so that we will be able to gather realistic infrared images as well as visual images for daytime missions.

A view of a section of the 109th airfield is shown in Figure 8. A number of the C130 aircraft can be seen as well as hangars adjacent to the runway. Just visible near the noses of each aircraft are electrical charging units. The general layout of the base an the surrounding area is shown in the high altitute aerial photo in Figure 9.

We now describe a number of experiments that are planned for the 109th site:

- Vehicle Counting – The base normally has a fixed number of resident aircraft. These aircraft can be away on a mission, shielded from view in a hangar, partially visible in a repair facility ("nose dock"), or completely visible on the runway apron. There are also various support vehicles, such as fuel trucks, which also have fairly well defined alternative locations. The first experiment will be to simply provide an inventory of the aircraft on the base in a sequence of images and maintain a consistent accounting of the sequence of vehicle movements.

- Fueling and Starting – The maintenance and operation of the aircraft are carried out in a relatively structured manner. Figure 10 shows a diagram of the required vehicle locations during the servicing of the aircraft [Operations].

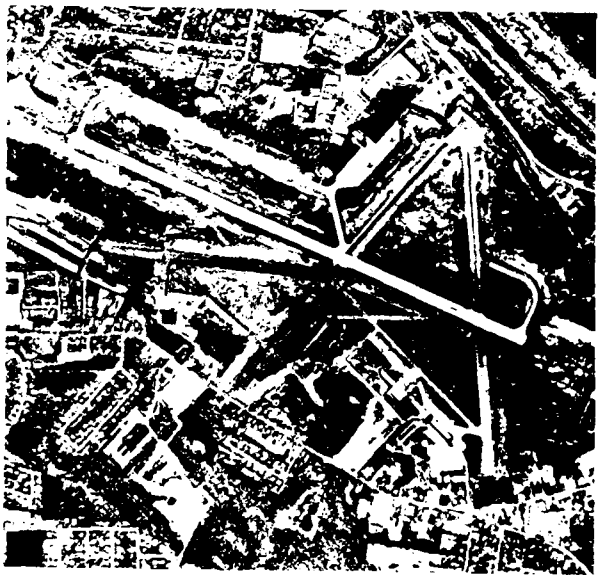


Figure 9: A high altitude view of the base.

The experiment will involve using PACE to recognize this operational configuration and reach a conclusion about the status of each aircraft on the apron. The experiment will include the case of cargo loading, which also has rather detailed operational specifications.

In both of these experiments, we will make use of extensive

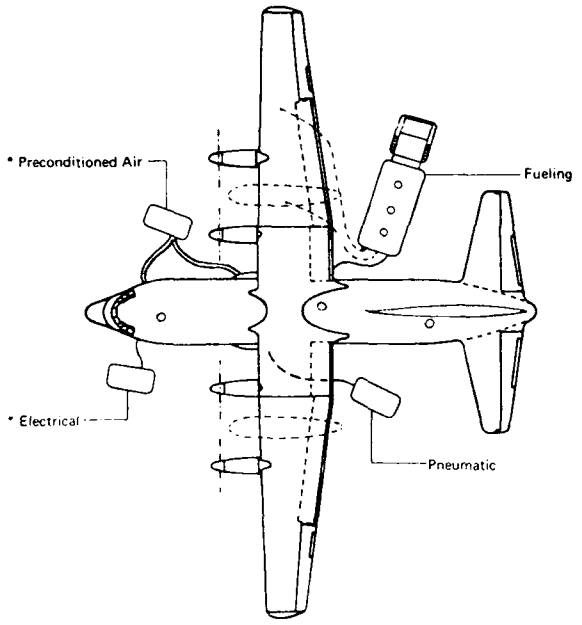



Figure 10: The configuration of service vehicles and support stations.

priori information about the base including detailed geometric models of the vehicles and buildings. We will also make use of relationships derived from natural language reports and messages. The linguistic analysis will be carried out using the TRUMP language interface, developed at GE [Jacobs].

## 4 Initial Results

We have already carried out matching experiments on the recognition of the C130 aircraft itself [Thompson and Mundy]. The generation of the data base of buildings, vehicles and terrain is currently underway. The resultant individual models will be merged into the world coordinate reference frame for the base. Images of the individual elements will be texture mapped onto the underlying models. The following figures show some of the elements of the data base.

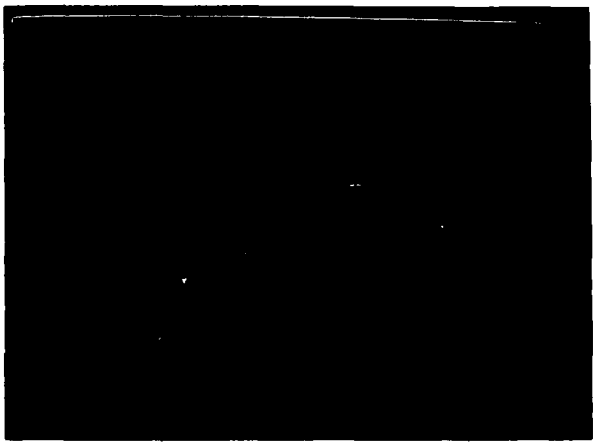


Figure 11: Main Hanger Model

Figure 12: Nose Dock Model

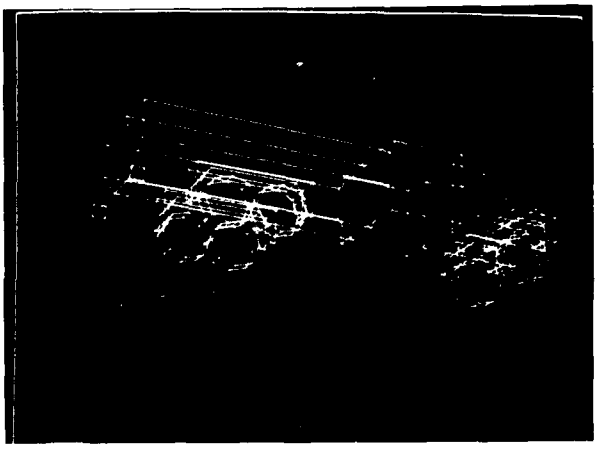


Figure 13: Fuel Truck Model

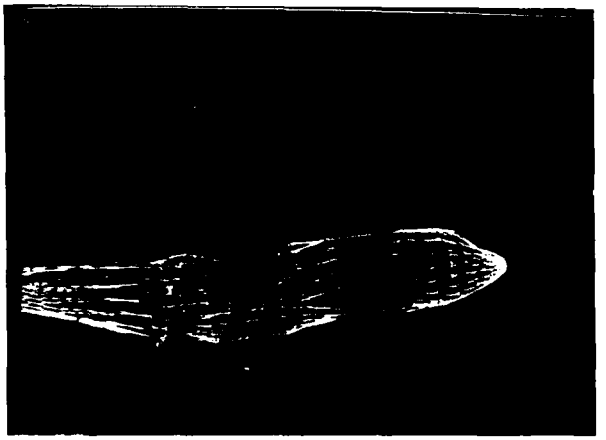


Figure 14: C 130 Model

## References


[Photogr.Man.] *Manual of Photogrammetry*, American Society of Photgrammetry, Falls Church,Va.,1980.

[Thompson and Mundy] Thompson, D., Mundy, J.L., "Three-Dimensional Model Matching From an Unconstrained Viewpoint", Proc. IEEE Robotics and Automation, 1987, pp. 280.

[Canny] "Finding Edges and Lines in Images", MIT Artificial Intelligence Laboratory Report, AI-TR-720.

[Asada and Brady] Asada, H. and Brady, M. "The Curvature Primal Sketch", Proc. IEEE Workshop on Computer Vision: Representation and Control, 1984.

[Connolly and Stenstrom] Connolly, C.I, Stenstrom, J.R., "Construction of Polyhedral Models from Multiple Range Views", Proc 8th International Conference on Pattern Recognition, 1986.

[Heller et al] "The Concept of an Effective Viewpoint," Mundy, J.L.,Heller, A.J., and Thompson, D.W., Proc. DARPA Image Understanding Workshop, Morgan Kaufmann, Los Altos,Ca., 1988.

[Mundy] Mundy, J. L. "Reasoning about 3-D Space with Algebraic Deduction", *Robotics Research, the Third International Symposium*, MIT Press, 1986 Cambridge, MA. pp. 117.

[Lowe] Lowe, D., "Perceptual Organization and Visual Recognition", Kluwer Academic Publishers, Boston, MA, 1985.

[Hanson et al] Hanson, A. J., Pentland, A. P., Quam, L. H., "Design of a Prototype Interactive Cartographic Display and Analysis Environment", Proc. Image Understanding Workshop, February 1987, Vol II, pp. 475.

[Smith and Strat] "Information Management in a Sensor-based Autonomous System," Proc. Image Understanding Workshop, Morgan Kaufman, Los Altos, Ca., 1987.

[Operations] "C-130H Operational Planning Data," Military Market Engineering, Lockheed-Georgia Company, Marietta, Ga., 1983.

[Jacobs] ,"A Knowledge Framework For Natural Language Analysis," Proc. 10th IJCAI, 1987

[Wesley and Markowsky] Markowsky, G., Wesley, G., "Fleshing Out Wire Frames", IBM Journal of Research and Development, Vol 24, 1980.

[Brown] Brown, C.M. "PADL-2: A Technical Summary " IEEE Computer Graphics and Applications, March 1982, pp. 69-84.

# AFFINE INVARIANT MATCHING

*Robert Hummel and Haim Wolfson*

Robotics Research Laboratory
Courant Institute, New York University
251 Mercer Street, New York, NY 10012

**Abstract**

We begin with a selected overview of computer vision research at New York University, emphasizing work on **representation of images**, and on **model-based vision**. We then focus on a technique for fast matching of shape descriptions of objects to preprocessed models. The matching process permits general affine transformations of the shape, and thus is applicable to the problem of recognizing flat, or nearly flat, objects viewed from an arbitrary angle and distance in 3-D. Different methods apply when the boundary of the objects consist primarily of line segments, or general (non-convex) curves, or (the most difficult case), convex curves. We also discuss the case of *recognizing* 3-D *objects, as opposed to flat objects.* The novelty of the approach and the computational efficiency is achieved by a hashing technique indexed over possible affine transformations.

## 1. A selected summary of Computer Vision research at NYU

In Table 1, we list a number of researchers, their titles and affiliations within NYU, and topics of research interest. Within recent years, computer vision research has become a speciality in the departments of computer science and psychology at New York University. Research in the psychology department has focused on computational vision studies through psychophysics and cell recordings, whereas research within the Courant Institute's computer science department has included high-level vision and sensor design within the Robotics Laboratory, and theories of low-level vision and mathematical analysis of feature extraction within the interdisciplinary "representation of images" sponsored research program. Most of the researchers listed in Table 1 independently direct a group of students, programmers, and other staff. We describe some of the activities of these groups below.

### 1.1. Representation of images

Professor Robert Hummel, in collaboration with Professor Michael Landy in the Psychology Department, has been directing a research project on low-level vision, in which they attempt to take a methodical, analytical, and mathematically sound approach to a comparative analysis of representations that can be useful for the analysis of images. The work is motivated by knowledge that human visual processing is mediated by center-surround receptive fields of various sizes in early stages of the visual pathway, and by directional and motion selective processing in subsequent stages. Accordingly, the research has focused on representational issues involving pyramid data structures, scale-space structures, and on methods for extracting information from low-level cues, such as depth from motion perception, and on methods for combining information from multiple cues to provide higher-level inferences.

Three methods are used to evaluate a representation. First, a proposed representation should be analyzed mathematically, to the extent possible. Questions of *completeness, continuity, and stability are the standard* mathematical questions that must be addressed. Next, the information content of a representation can be assessed by using psychophysics to determine the information that is used by the vision system to induce perceptual effects. For example, we can produce a reconstruction, and conduct experiments to compare the information content in an original image with the impoverished reconstruction. Finally, a representation can be shown to be worthwhile if a useful vision system can make use of the construct. We next describe how several subprojects support this methodology of representation evaluation.

#### 1.1.1. Solving ill-conditioned inverse problems

Many problems in vision involve the solution of inverse problems. Interpolation, segmentation, motion extraction, shape-from-shading and other cues, and many other standard problems in vision can be viewed as inverse problems. Moreover, the problem of reconstructing image data from a representation is also frequently an inverse problem. At the Courant Institute, interest in inverse problems is a hallmark of research work over the past fifty years.

Nearly all inverse problems are ill-posed, in the sense that either *no solution* exists, or many solutions exist, or if one exists, there are many other candidate solutions which are nearly as good. The latter situation, when many images project to nearly the same representation, characterizes the case of an ill-conditioned inverse problem. A prototypical example is the deblurring problem: to recover the image data from a Gaussian-blurred version of the original image. In [1], we examined a novel filtering approach

| Researcher | Position | Vision Interests |
|---|---|---|
| J. Schwartz | Professor, Computer Science (on leave) | Model-based vision, matching |
| M. Sharir | Professor, Computer Science (visiting) | Shape representation |
| R. Hummel | Assistant Professor, Computer Science | Representations, feature extraction, knowledge aggregation |
| H. Wolfson | Research Scientist, Computer Science | Efficient matching, model-based vision |
| P. Wright | Professor, Computer Science | Automated manufacturing |
| E. Schwartz | Adjunct Professor, Computer Science Professor, Medical Center | Neurophysiology, retinal mapping, extra-striate physiology |
| D. Lowe | Assistant Professor, Computer Science (on leave) | Model-based vision |
| M. Bastuscheck | Research Scientist, Computer Science | 3-D sensor design |
| J. Hong | Professor, Computer Science (visiting) | Shape representation |
| X. Tan | Research Scientist (visiting) | Shape matching |
| M. Landy | Assistant Professor, Psychology | Shape perception |
| R. Shapley | Professor, Psychology | Single cell recordings, visual pathway |
| A. Movshon | Associate Professor, Psychology | Neuroanatomical mapping, motion |
| G. Sperling | Professor, Psychology | Psychophysics: Motion perception, attention, computational vision |

**Table 1.** Selected NYU Researchers in Computer Vision

to deblurring. More practical instances of vision-related ill-conditioned inverse problems include reconstructions from zero-crossings, interpolation, and reconstructions from sparsely sampled filtered data. These topics have been addressed in subsequent studies.

One way to approach certain ill-conditioned inverse problems, formulated as variational minimization problems, is to add a regularization term. When done carefully, the result is an image or function that is not quite a solution to the original problem, but satisfies some smoothness constraints. By allowing for *controlled continuity*, it is even possible to allow for some discontinuities.

A major contribution of research effort by Professor Hummel and Dr. Moniot has been to show the utility of a different approach. The method, called *minimization of equation error*, is applicable whenever the inverse problem involves the specification of information in scale-space. To date, the technique has been applied to deblurring [2] and to reconstructions from zero-crossings [3]. The results in both domains have been excellent. A principle advantage of the approach is that there is no explicit smoothness constraint. Instead, there is an assumption that the supplied information came about from a sampling in scale-space.

### 1.1.2. Scale-space representations

In a number of publications and presentations, researchers at NYU have become major proponents of the scale-space viewpoint on pyramid data structures [4, 5]. The idea of using the Heat Equation to model the process of representing image data by a continuum of Gaussian-blurred images is not new, but has become the basis for much of our mathematical analysis of related representations. In particular, we have shown that the "evolution property of zero-crossings," the famous property that says that in scale-space zero-crossing contours are never created as one passes from fine to coarse resolution, is mathematically equivalent to the Maximum Principle for parabolic equations [5].

Zero-crossings in scale-space have become a popular proposed representation, especially since the zero-crossings are correlated with edge information. One difficulty with zero-crossings is that they do not vary continuously with the image data; a second problem, as we have established, is that the representation is unstable. Nonetheless, by working on reconstructions from zero-crossings, we have been able to establish that the information carried in the zero-crossings is a rich description of

image information, and that the reconstructions contain many recognizable features of the original data. This suggests that the zero-crossings enhanced with slightly more information should suffice for a good representation.

Indeed, in more recent experiments, we have established that stable reconstruction is possible when zero-crossings are enhanced with gradient data along the zero-crossings. The fact that such a representation is complete was conjectured by Marr, established theoretically by us in a paper a couple years ago, but only established in stable numerical experiments by us this past summer.

It is also possible that the structure can be simplified to make for a more useful image representation, using alternatives to zero-crossings altogether. Specifically, the visual cortex seems to have cells that respond to local bandpass filters of visual data, and to directional derivatives of that data. By sampling data and derivatives of data in a Laplacian-of-Gaussian scale-space, we believe that an effective representation can be constructed. Others have similarly been concentrating on "oriented pyramids," for example using the "wavelet" transformation, and we find much of this work to be persuasive. We expect to use our reconstruction techniques and mathematical analytical methods to study these and related representations. Ultimately, we seem to be focusing on a representation involving sampling in scale-space, and cortical computations for reconstruction of a representation that encodes the complete Laplacian pyramid.

### 1.1.3. Kinetic depth effect and depth perception

Another way to probe the information that is retained by our image analysis system is to determine the cues that are important for certain perceptual effects. Professor Michael Landy, in collaboration with postdocs and students, has extended understanding of the kinetic depth effect, by conducting many experiments varying motion cues and evaluating depth perception. A key to this work is our ability to measure the perceptual effectiveness of the kinetic depth effect in particular psychophysical experiments, based on a collection of three-dimensional shapes. It is important to isolate the cues presented to subjects in order to assess independently the information that is used in the computation of shape parameters. For example, we are able to remove local dot density as a cue, and still retain shape identification from moving dots. Further, we have shown that while visual motion is the essential input stimulus that is responsible for the kinetic depth effect, the optic flow perception need not be computed by a Fourier energy detection system, such as the Reichardt model. Moreover, pairs of features do not seem to be important in depth perception. We conclude that the kinetic depth effect is based on a representation of optical flow that is preattentive and based on global computations. We are thus developing a model of optic flow computation feeding a model for generating a kinetic depth effect [6].

Another interest has related to depth perception from stereo. The cooperative psychology and computer science group has begun to develop a multiresolution model of stereo perception. However, the model suggested a number of experiments for psychophysical investigation, and these experiments have been conducted by students in our Psychology department. At interest is the kinds of interpolations that are used when the stereo information is sparse. By discovering that smooth interpolants seem to dominate, we have constrained models for stereo perception.

### 1.1.4. Knowledge aggregation

A large body of research in AI is concerned with the combination of knowledge from different sources of information. In computer vision, cues can come from edge data, texture, orientation, dynamic processing of temporal data, color, models, and other sensory sources. To make inferences from disparate sources of knowledge requires a representation of information in a fashion that permits incremental modification.

We have long been associated with relaxation labeling methods for image analysis [7]. We have developed a number of relaxation labeling models, and continue to show their effectiveness at handling low-level visual tasks. However, there are many related alternatives to relaxation labeling, and we have pointed out relationships between relaxation labeling, stochastic relaxation methods, brain state models, neural networks, and the Dempster/Shafer theory of evidence [8].

The latter topic, the "theory of evidence," has a large and dedicated set of advocates in the AI community. By bringing a mathematical and statistical viewpoint to this field, and by contrasting the methods involving the theory of evidence to other knowledge aggregation methods, we have been able to explain the foundations of the Dempster rule of combination, and show that the basis of the formula is Bayesian combination of opinions, where the state of the system is represented by the statistics of more than one opinion. Using this viewpoint, we are able to suggest alternative formulations, which end up looking remarkably like Kalman filtering [9, 10]. One of our extensions, presented at the last IJCAI meeting, incorporates a notion of parameterized independence, relaxing normal assumptions of complete (conditional) independence.

## 1.2. Computational neuroscience

Professor Eric Schwartz directs a large group concerned with computational neuroscience. Using studies of the visual cortex of monkeys, complex patterns of functional areas have been identified. In this project, we are especially concerned with the mappings along the visual pathway, and the algorithms that are suggested by the data structures created by the precise organization of information. Attempts at understanding the nature of visual cortex pose a wide range of problems in computer graphics, image processing, computational geometry, and numerical methods. In a series of studies, computer graphics and image processing methods have been used to develop accurate three-dimensional models of the retinotopic map, and to represent these maps by numerically flattening layers using a method of minimal metric error [11, 12]. Using results obtained from these methods and dioxyglucose studies, subsequent work has suggested a number of algorithmic methods that arise from the functional neuroanatomy. In particular, a novel computational method for stereopsis has been suggested based upon the striations in primary visual cortex [13, 14]. A shape representation scheme has also been presented, and recent work centers

on pattern recognition interpretations of cortical functional maps.

### 1.3. Representing shape

In addition to studying representations of grayscale image data, members of the Courant Institute Robotics Laboratory have been concerned with more symbolic specifications of image constructs. In particular, we have been concerned with the specification of two and three-dimensional shape information. In this study, our principle evaluation criterion for a shape representation is its effectiveness in an object recognition system.

For description of 3-D shapes in a manufacturing environment, it is reasonable to assume the existence of 3-D depth data, obtained from a depth sensor. In this regard, the NYU Robotics Laboratory has pursued the development of a novel light-striped depth sensor, yielding simultaneous intensity and range data [15,16]. Using depth data extracted by this depth sensor, and also by the laser-based "White Scanner," descriptions of 3-D objects have been developed.

For 2-D object description, Yaron Menczel in his thesis work demonstrated that shape information can be effectively encoded by a graph structure that is derived from the orientation of boundary points [17]. Specifically, each point in a region is labeled with an orientation tag based on the orientation of the nearest boundary point, and the orientation field is quantized to produce regions of similarly labeled points. The resulting graph is used for matching and recognition, and proved successful for applications such as character recognition with variable fonts and connected letters.

A principle motivation in the study of shape information is that shape identification is possible in the presence of occlusion and obscuration. Thus it is evident that shape analysis for this purpose should be local, enabling partial matching techniques. Since 2-D objects are fully described by their boundary curves, both globally and locally, and 3-D objects may also be represented by sets of *characteristic curves*, (e.g. ridges, curves of sharp intensity change, curves of specularity), considerable effort has been done to develop efficient curve representation and matching algorithms both in 2-D and 3-D (using range data, as described above). This effort was initiated by the work of Professors J. T. Schwartz and M. Sharir and their co-workers, and has been carried on and expanded in the last two years by Dr. H. J. Wolfson and graduate students associated with the Lab [18-22]. A landmark in application of this method in the 2-D case, and a clear demonstration of its sensitivity and robustness, was its use to assemble (graphically rather than physically) all the pieces of two intermixed hundred-piece commercial jigsaw puzzles from separate photographs of the individual pieces [23].

The work is founded on a new technique for geometrically hashing two-dimensional and three-dimensional curves [24-26]. Curves are represented by local features that are invariant to rigid transformations, and feature values are used to generate an attribute of a curve called its *footprint*, which enables us to efficiently index the appropriate local information to the object for recognition purposes.

This effort has been recently extended to local representation of 2-D curves in an affine invariant way [27]. We will elaborate on this issue in Section 2.

In a separate but related project, some particularly elegant work on affine invariant *global* shape representation has been completed recently by Professor J. Hong and Dr. X. Tan, who are currently visiting our Lab [28].

Finally, we have applied an interest in parallel algorithms to shape analysis. Much work in parallel image processing focuses on SIMD architectures, where many simple processors perform the same function distributed over an image. However, it is clear that in biology, there is a diversity of function in massive parallelism, so that models of multiple program streams (MIMD) are in a sense more realistic. We have done a limited amount of work on connected component algorithms for image analysis, focusing on issues of MIMD parallelism [29,30]. NYU has a large group of researchers involved in parallel algorithm development, parallel architecture studies, and parallel system design. In particular, the "Ultracomputer" project works closely with IBM's RP3 project to develop an MIMD shared-memory machine with a combining network, together with an highly parallel operating system based on Berkeley UNIX. Many image processing and vision research projects (for example, the matching algorithms based on the *footprint* technique), will be greatly facilitated by the accessibility of this unique parallel machine.

### 1.4. Model-based vision

Our work on representation and description of scenes has led to a large program of work on object recognition using the techniques of model-based vision. A number of methods progressing along complementary lines have been developed.

The curve matching techniques using *footprint* representations, which were mentioned in the previous section, led to the development of an experimental 2-D object recognition system enabling us to recognize overlapping 2-dimensional objects selected from large databases of model objects without significant performance degradation as the size of the data base increases [24]. Experimental results from databases of size about 100 make this technique appear quite promising.

A complete, working model-based system, SCERPO, was developed by Professor David Lowe while at NYU [31]. Several graduate students at NYU continue working on this system, under direction of Professor Lowe (although David has now returned to the University of British Columbia). For example, Robert Goldberg is extending SCERPO to the case where the models have articulation joints [32].

The existing, functioning, SCERPO system is one of the first to demonstrate the recognition of three-dimensional objects in single images taken from arbitrary viewpoints, incorporating fast matching methods using a grouping strategy. The objects are represented as polyhedral solids. Features are extracted from the scene by means of low-level edge feature extraction [33], followed by simple grouping and feature description operations. Matching is done by solving for the three-dimensional

position and orientation of an object directly from two-dimensional image measurements, using an iterative, hill-climbing technique to find the best viewpoint parameters that will "project" the object model onto the locations of matched image features. An important point is that the evaluation of the quality of a match, is done in the projected, two-dimensional image space. Thus methods to recover 3-D object shape from image cues are unnecessary. Once a few initial matches have been formed, it can make quantitative predictions for the exact locations of further model features in the image. This provides a reliable method for evaluating the correctness of a match according to whether it is consistent with a single viewpoint [34].

A second aspect of the SCERPO research concerns the problem of perceptual organization. Human vision is able to detect many different types of significant groupings of image elements, such as parallelism, collinearity, proximity, or symmetry in an otherwise random set of image features. This perceptual organization capability has been missing from most computer vision systems. Since these image groupings reflect viewpoint-invariant aspects of a three-dimensional scene, they are ideal structures for bridging the gap between the two-dimensional image and the three-dimensional model. Probabilistic measures have been developed for evaluating the significance of instances of each of these image relations that can arise from projective invariance. The SCERPO system uses these significance measures to prioritize prototype matches for object recognition [35].

Yet another project in model-based object recognition from single 2-D images is being directed by Dr. Haim Wolfson. This system is based on affine invariant point, line, and curve matching, and uses the affine approximation of the viewing transformation to facilitate efficient matching procedures. The system will be able to deal efficiently with both polyhedral and non-polyhedral scenes with considerable occlusion. Some of the algorithms have been already successfully tested in recognition of flat objects in 3-D scenes from an arbitrary viewpoint [27, 36], and it is currently being extended to enable recognition of general 3-D objects. This work will be addressed in detail in Section 2.

## 2. Efficient matching

We now present a design and results of work by H. Wolfson, in collaboration with H. Lamdan, and J. Schwartz, on object recognition in the presence of arbitrary affine transformations of the models.

We develop new techniques for model-based recognition of 3-D objects from unknown viewpoints. The method is especially useful for recognition of scenes with overlapping and partially occluded objects. An efficient matching algorithm, which assumes affine approximation to the perspective viewing transformation, is proposed. The algorithm has an off-line model preprocessing phase and a recognition phase to reduce matching complexity. The algorithm has been successfully tested in recognition of flat industrial objects appearing in composite occluded scenes.

### 2.1. Introduction

Recognition of industrial parts and their location in a factory environment is a major task in robot vision. Most practical vision systems are model-based systems (see the survey in [37]). Object recognition using model-based vision presents many challenges in image understanding, but offers the possibilities of well-formulated tasks and rigorous algorithm evaluation.

We consider the object recognition problem, where the vision system is faced with a composite scene of overlapping parts (thus partially occluding each other), taken from a data-base of known objects. The task is to recognize the objects in the scene and to specify their location and orientation.

No restriction on the viewing angle of the camera is assumed. We begin by considering the recognition of flat objects arbitrarily positioned in space. At the end of this section, we discuss the use of these methods for the general case of 3-D objects. The recognition is done from 2-D intensity images. The algorithms that we describe have been actually tested for the recognition of objects comprising composite scenes of industrial tools, such as pliers, wrenches, etc., (see Figs. 4-8).

Since we are concerned with recognition of partially occluded objects, the use of global features is precluded. Accordingly, we must describe our objects by a set of local features. This same conclusion is applicable to the human vision system, which is also capable of recognition in the presence of considerable occlusion. The local features can be points, line segments, curve segments, borders, or other structures developed from local description operations. Initially, we restrict ourselves to the use of special points, which we denote as *interest points*. The point sets of the various model objects are matched against the point set of the composite overlapping scene using a small number of corresponding points. Once a prototype correspondence is established, we find the best transformation in least-squares sense to establish the correct position of the model object in the scene image. A key aspect of our scheme is its computational efficiency, based upon a division into a preprocessing stage and a recognition stage. Our model point sets are preprocessed off-line independently of the scene information, thus enabling an efficient on-line recognition stage. A major advantage of the proposed matching algorithm is the ease with which both the preprocessing and recognition stages can be parallelized.

The problem of object recognition in 2-D scenes is a common one [24, 38-41]. Three-dimensional object recognition systems are discussed in [37, 42]. Recent image understanding results not mentioned in the above surveys include [31, 43, 44].

The method described here differs from other existing model-based matching systems. Our method, which uses a hashing scheme indexed on the affine transformation and model type, is more algorithmic and more parallelizable than Lowe's SCERPO system [31]. In [44], a clustering approach is used to discover the transformation between the model and the scene images. The hashing scheme here

is more efficient and more predictable. In [43], there is an emphasis on the classification of the model and image features to reduce the complexity of matching, while the matching algorithm itself is straightforward. We, on the other hand, consider the case where no such effective classification can be done (this is also the assumption in [44]), and, hence, our emphasis is on the development of an efficient feature matching algorithm, which processes the models and the scene images independently allowing fast recognition. In case feature classification is possible it can be incorporated in our algorithm in a natural way to improve its efficiency.

## 2.2. Definition of the Problem

We initially assume that we view partially occluded flat objects from an arbitrary viewpoint. These initial assumptions are similar to those in [43]. We also assume that the depth of the centroids of the objects in the scene is large compared to the focal length of the camera, and that the depth variation of the objects are small compared to the depth of their centroids. Under these assumptions it is well known that the perspective projection is well approximated by a parallel (orthographic) projection with a scale factor (see for example p.79 in [45]). Hence, two different images of the same flat object are in an affine 2-D correspondence: namely there is a non singular $2\times2$ matrix $\mathbf{A}$ and a 2-D (translation) vector $\mathbf{b}$, such that each point $\mathbf{x}$ in the first image is translated to the corresponding point $\mathbf{Ax} + \mathbf{b}$ in the second image.

Our problem is to recognize the objects in the scene, and for each recognized object to find the affine transformation that gives the best least-squares fit between the model of the object and its transformed image in the scene.

## 2.3. Choice of 'Interest Points'

The matching algorithm, which is described in the next section, is based on matching 'interest points', extracted in both the scene image and the model. These should be database dependent, so that different databases of models will suggest different features for 'interest points'. For example, a data base of polyhedral objects naturally suggests the use of polyhedra vertices as 'interest points', while 'curved' objects suggest the use of sharp convexities, deep concavities and, maybe, zero curvature points. 'Interest points' do not have to appear physically in the image. For example, a point may be taken as the intersection of two non-parallel line segments, which are not necessarily touching. An 'interest point' does not necessarily have to correspond to a geometrical feature. The Moravec 'interest operator', based on high variance in intensity, is described in [46] and was used in [47].

The problem of extracting stable, useful 'interest points' is a delicate topic equivalent, in many ways, to the shape representation problem. Although a successful approach to this problem is required for any model-based vision system, we will assume here that a sufficient number of stable points can be extracted from the relevant images. Our emphasis then, in this paper, is on the matching problem, and not on the model representation or image description.

In our experiments with 2-D objects, we used points of sharp convexities and deep concavities along the borders (see Figs. 4c, 4d, 5b, 6b).

## 2.4. Recognition of a Single Model in a Scene

For the sake of clarity we describe our algorithm in the simpler situation, where the database consists only of one model. However, the presentation given here applies to the general case where a number of models may appear in the scene.

It is well known that an affine transformation of the plane is uniquely defined by the transformation of three non-collinear points (see, for example, [48]). Moreover, there is a unique affine transformation, which maps any non-collinear triplet in the plane to another non-collinear triplet. Hence, we may extract "interest points" on the model and the scene, and try to match non-collinear triplets of such points to obtain candidate affine transformations. Each such transformation can be checked by matching the transformed model against the scene. This is also the basic approach in [43].

However, the complexity of such a scheme is quite unfavorable. Given $m$ points in the model and $n$ points in the scene, the worst case complexity is $(m\times n)^3 \times t$, where $t$ is the complexity of matching the model against the scene. If we assume that $m$ and $n$ are of the same magnitude, and $t$ is at least of magnitude $m$, the worst case complexity is of order $n^7$. One way to reduce this complexity ([43] ) is to classify the points in a distinctive way, so that each triplet can match only a small number of other triplets. We consider, however, the situation were such a distinction does not exist or cannot be made in a reliable way (see [44]). Hence, we present a more efficient triplet matching algorithm. Our method has the advantage that when distinguished points are classifiable, or when the transformation can be restricted to a smaller class, the complexity will be reduced.

The algorithm consists of two major steps. The first one is a preprocessing step which is applied to the model points. This step does not use any information about the scene and is executed off-line before actual matching is attempted. The second step, matching proper, uses the data prepared by the first step to match the models against the scene. The execution time of this second step is the actual recognition time.

In order to separate the computation into two such independent steps, we have to represent the model and scene point information in a way that is both independent and still allows comparison of corresponding structures.

The crucial observation is that once an affine basis is specified by a triplet of non-collinear points, then the coordinates of all the other points, given in the coordinate system of the triplet, are affine invariant. That is, if $\mathbf{e}_{10}$, $\mathbf{e}_{01}$, and $\mathbf{e}_{00}$ are three non-collinear points, then any other point $\mathbf{v}$, with coordinates $(\xi,\eta)$:

$$\mathbf{v} = \xi(\mathbf{e}_{10} - \mathbf{e}_{00}) + \eta(\mathbf{e}_{01} - \mathbf{e}_{00}) + \mathbf{e}_{00}$$

will still have coordinates $(\xi,\eta)$ if the entire figure is

translated by the affine transformation T:

$$\mathbf{Tv} = \xi(\mathbf{Te_{10}} - \mathbf{Te_{00}}) + \eta(\mathbf{Te_{01}} - \mathbf{Te_{00}}) + \mathbf{Te_{00}}$$

assuming the same triplet of points, now $\mathbf{Te_{00}}$, $\mathbf{Te_{10}}$, $\mathbf{Te_{01}}$ are chosen as the basis.

Accordingly, our data structure for representing a given object will be based on a collection of mappings from the set of all non-collinear triplets into a list of quantized coordinate pairs. Actually, we will form a hash table, where each quantized coordinate pair, i.e., a box represented by $(\xi,\eta)$, contains a list of all object models and their basis triplets that contained an interest point that mapped to that box.

Our algorithm will efficiently compare these sets of coordinates belonging to different bases. The algorithm is as follows:

**(A) Preprocessing**

Assume we are given an image of a model, where $m$ 'interest points' have been extracted. For each ordered non-collinear triplet of model points, the coordinates of all other $m-3$ model points are computed taking this triplet as an affine basis of the 2-D plane. Each such coordinate (after a proper quantization) is used as an entry to a hash-table, where we record the number of the basis-triplet with which the coordinates were obtained and the number of the model (in case of more than one model). The complexity of this preprocessing step is of order $m^4$ per model. New models added to the data-base can be processed independently without recomputing the hash-table.

**(B) Recognition**

In the recognition stage we are given an image of a scene, where $n$ 'interest points' have been extracted. We choose an arbitrary ordered triplet in the scene and compute the coordinates of the scene points taking this triplet as an affine basis. For each such coordinate we check the appropriate entry in the hash-table, and for every pair *(model, basis-triplet)*, which appears there, we tally a vote for the model and the basis-triplet as corresponding to the triplet which was chosen in the scene. (If there is only one model, we have to vote for the basis triplet alone).

If a certain pair *(model, basis-triplet)* scores a large number of votes, we decide that this triplet corresponds to the one chosen in the scene. The uniquely defined affine transformation between these triplets is assumed to be the transformation between the model and the scene. If the current triplet does not yield a model and triplet that scores high enough, we pass to another basis-triplet in the scene.

For the algorithm to be successful it is enough, theoretically, to pick any three non-collinear points in the scene belonging to one model. The voting process, per triplet, is linear in the number of points in the scene. Hence, the overall recognition time is dependent on the number of model points in the scene, and the number of additional 'interest points' which belong to the scene, but did not appear on any of the models. Although, in the worst case, we might have an order of $n^4$ operations, in most cases, especially when the number of models is small, the algorithm will be much faster. For example, if there are $k$ model points in a scene of $n$ points, then the probability of not choosing a model triplet in $t$ trials is approximately

$$p=(1-(\frac{k}{n})^3)^t$$

Hence, for a given $\epsilon>0$, if we assume a lower bound on the 'density' $d=\frac{k}{n}$ of model points in a scene, then the number of trials $t$ giving $p<\epsilon$ is of order $\frac{\log\epsilon}{\log(1-d^3)}$, which is a constant independent of $n$. Since the verification process is linear in $n$, we have, in this case, an algorithm of complexity $O(n)$, which will succeed with probability of at least $1-\epsilon$.

This method assumes no *a-priori* classification of the model and scene points to achieve matching candidates. If such information is available, it can be incorporated into our method by assigning weights to the correspondence of different triplets to the model, and by checking the triplets in an appropriate order.

Numerical errors in the point coordinates are more severe when the basis points are close to each other compared to the other model points in the scene. To overcome this problem, we may introduce the following procedure. If a certain basis triplet gets a number of votes, which, on one hand, are not enough to accept it as a 'candidate' basis, but, on the other hand, do not justify total rejection, we may change this triplet to another triplet consisting of points that were among the 'voting' coordinate pairs, and are more distant from each other than the previous basis points. In the correct case this procedure will result in a growing match, as the numerical errors become less significant. Even if a basis-triplet belonging to some model did not get enough votes due to noisy data, we still have chance to recover this model from another basis-triplet.

A major potential advantage of the suggested algorithm is its high inherent parallelism. Parallel implementation of this algorithm is straightforward.

## 2.5. Finding the Best Least-Squares Match

Suppose that for a particular basis triplet chosen in the scene, a high number of votes are obtained for a given *(model, basis-triplet)*. Each vote implies the existence of a match (by close proximity) of an interest point in the scene with some point in the specified model. This match is valid for the affine transformation that maps the basis triplet in the scene to the basis triplet of the model receiving many votes. We can then improve this affine transformation, and potentially find more matches by finding the optimal affine transformation for the set of matched points. This is efficiently accomplished if the measure of optimality is the sum of square distances in errors of the match (details are given in [ 36]). Other measures are also possible.

We incorporated this process of affine transformation improvement in our experiments. In Fig. 6c we see an example of a fit obtained by calculating the affine transformation from three basis points, and in Fig. 6d the same model is fitted using the best least-squares affine match, based on 10 points, all of which, by the way, were recovered as corresponding points by the transformation in Fig. 6c.

## 2.6. Summary of the Algorithm

Our algorithm can be summarized as follows:

(A) Represent the model objects by sets of 'interest points'.

(B) For each non-collinear triplet of model points compute the coordinates of all the other model points according to this basis triplet and hash these coordinates into a table which stores all the pairs *(model, basis-triplet)* for every quantized coordinate pair.

(C) Given an image of a scene extract its interest points, choose a triplet of non-collinear points as a basis triplet and compute the coordinates of the other points in this basis. For each such quantized coordinate pair, vote for the pairs *(model, basis-triplet)*, that appear in the hash table at that location and find the pairs which obtained the most number of votes. If a certain pair scored a large number of votes, decide that its model and basis triplet correspond to the one chosen in the scene. If not, continue by checking another basis triplet.

(D) For each candidate model and basis triplet from the previous step, establish a correspondence between the model points and the appropriate scene points, and find the affine transformation giving the best least-squares match for these corresponding sets. If the least-squares difference is too big go back to Step (C) for another candidate triplet. Finally, the transformed model is compared with the scene (this time we are considering not only previously extracted 'interest points'). If this comparison gives a bad result go back again to Step (C). (In our experiments we compared the boundaries of our objects at equally spaced sample points.)

This is a short summary of the basic algorithm. Of course, various improvements can be incorporated in its different steps. We discuss a number of possibilities in the next section.

## 2.7. Reduction of Complexity using Affine Invariants

When the number of 'interest points' on the models is large, various affine invariants can be exploited to reduce the complexity of the method presented in Section 4. We give one such example. We will use the following observation (see, for example, p.73 in [45]). Two straight lines which correspond in an affine transformation are 'similar', i.e. corresponding segments on the two lines have the same length ratio. The same statement holds for sets of parallel lines. Hence, if we have a set of points, which are located on parallel lines in a model, and another set of points on parallel lines in the scene, we can efficiently check the conjecture that some of these points correspond.

Let us see how the previous method can be modified for the case when the 'interest points' lie on a collection of lines. We have again two major steps.

**(A) Model Preprocessing**

Extract the 'interest points' on the model and group the points into a collection of lines. (A point may belong to different lines.) Take an ordered pair of points on a line and compute the coordinates of all other model points on this line taking this pair as the standard one-dimensional basis of the line. Each such coordinate is used as an entry to a hash-table, where we record the number of the basis-pair at which the coordinate was obtained, the number of the line, and the number of the model.

**(B) Recognition**

Extract sets of points positioned on the same line in the image. Choose a pair of points on such a line as a basis and compute the coordinates of the other points on the same line according to this basis. For each such coordinate check the appropriate entry in the hash-table, and vote for every triple of *(model, line, basis-pair)*, which appears there. A triple that scores a large number of votes gives the correspondence between the points on the appropriate lines. Correspondence of three non-collinear points (obtained from different lines, of course), already gives a full affine basis, and we proceed as before.

The worst case complexity of this approach is less by a factor of $n$, since we now are iterating over pairs of points, as opposed to triples of points in the scene. The expected complexity is also less, since we are more likely to choose a pair of points belonging to a single model, over choosing a triplet within a single model.

A different reduction in complexity occurs if we are confronted with the problem of recognition of objects that have undergone a similarity transformation, i.e., rotation, translation, and scale. This is the situation when the viewing angle of the camera is the same both for the model and the image of a scene. Such conditions can be achieved, for example, in a factory environment where the viewing angle of a camera on a conveyor belt can be kept constant.

Our algorithm is obviously applicable without modification to the case of a similarity transformation, since it is a special case of an affine transformation. However, the complexity of both the preprocessing and recognition stage can be reduced. The key observation here is that since the similarity transformation is orthogonal, two points are enough to form a basis which spans the 2-D plane. (The first point is assigned coordinates (0,0) and the second (1,0). The third basis point (0,1) is uniquely defined by these two points.) Hence, we may repeat the procedure described in Section 4 using basic pairs instead of basic triplets. This reduces the complexity of the preprocessing step by a factor of $m$, and the worst case complexity of the recognition step by a factor of $n$.

## 2.8. Line Matching

In the previous sections we dealt with point matching algorithms. However, extraction of points might be quite noisy. A line is a more stable feature than a point. Thus in scenes were lines can be extracted in a reliable way, e.g. scenes of polyhedral objects, we might be interested to apply similar procedures to lines.

All the point matching techniques given above apply directly to lines, since lines can be viewed as points in the dual space. Thus three lines that have no parallel pairs are a basis of the affine space; each line has unique coordinates in this basis, and we repeat exactly the same matching procedure. We can also make use of line segments to reduce the complexity of the point matching when lines can be stably extracted from the scene. We omit the details here.

## 2.9. Curve Matching

In this section, we extend the methods of the previous sections to the case where the extracted features are no longer simple 'interest points,' but instead are entire boundary curves. Since the shape of planar rigid bodies is completely described by their boundary, object recognition can be accomplished by matching these curves. Matching of curves that have undergone affine transformation was discussed in the works of Cyganski and Orr ( [49,50]). Another elegant global curve matching method was recently developed by J. Hong and X. Tan [28]. Their methods, however, require knowledge of the full curve, and hence are unable to deal with occlusion. The method described in this section is based on local affine invariant features enabling recognition of partially occluded objects.

The curves are conveniently represented by vertices of their polygonal approximations. Ideally, the extraction includes a smoothing process, such as the one described in [18]. We discuss separately the cases of non-convex and convex curves.

### 2.9.1. Non-convex Curve Matching

As was pointed out in [51], non-convex planar regions are sometimes easier to handle than convex regions. In the case of an affine transformation, each concavity supplies us with a stable feature from which the affine transformation can be recovered. Specifically, consider the sketch of Fig. 1. The concavity depicted there is bounded by a single segment of the convex hull which we call the *concavity entrance*. It is a simple geometric observation that the concavity entrance is invariant under affine transformation. An additional point which is invariant under affine transformations is the concavity point most distant from the concavity entrance line. (If this point is not unique, we may choose the leftmost.) Thus, one can extract a concavity-based point triplet which is affine invariant. This basis triplet can be used in a recognition scheme similar to the point matching scheme.

The concavity entrances are computed as follows. First the convex hull of a polygonal approximation of the boundary curve is computed. The concavity entrance endpoints are those convex hull point pairs which are separated by polygon vertices not belonging to the convex hull. The computation of the (leftmost) boundary point

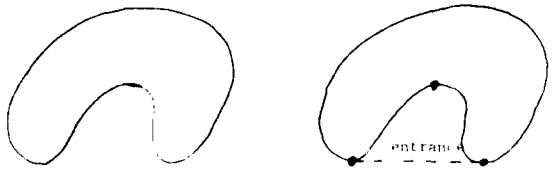


**Figure 1. A concavity entrance and basis triplet**

most distant from the concavity entrance is simple. The complexity of the entire process is $O(n)$, where $n$ is the number of polygon vertices (see [52], p.93).

The procedure for recognition of partially occluded non-convex objects in composite scenes may proceed exactly in the same way as as in the previously described point matching algorithm (see Section 2.6). Here, however, the the complexity is highly reduced, since we consider only basis triplets that are *concavity*-based. Moreover, since concavities may be differentiated by their shape even in the affine invariant case, we may further reduce the complexity of the algorithm by comparing only *basis triplets* based on affine invariantly similar concavities. To accomplish it we introduce a numerical affine invariant *shape characteristic* that we call a *footprint*. The footprint should be a continuous, stable, and easily computed representation of the concavity shape. To compute the footprint, we first normalize a concavity by applying the transformation which maps its triplet basis to a standard equilateral triangle. That is, the concavity endpoints are mapped to $(-1,0)$, $(1,0)$, and the third point to $(0,\sqrt{3})$. To each such normalized shape we assign a vector of numbers that we call the 'footprint.' One of the footprint schemes that we use is illustrated in Fig. 2. For some constant $s$ (say $5 \le s \le 10$ ), we divide the upper half plane by $s+1$ rays based at the origin, with angle $\frac{\pi}{s}$ between two consecutive rays. Let $a_i$ be the area of the 'normalized' shape between rays $i$ and $i+1$. The footprint will be $s$-vector $(a_1, a_2, \cdots, a_s)$, where each component is quantized into one of a number of discrete bins.

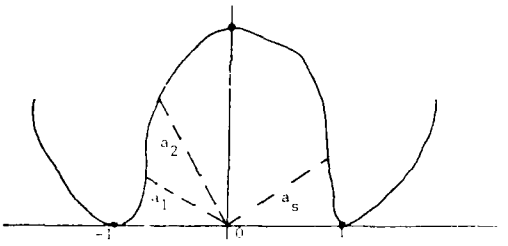



**Figure 2.** The footprint of a concavity

We now proceed as before, constructing a hash table. Each footprint is used as an entry to the hash table, where the *model* and *concavity* numbers are recorded. In the recognition phase, each concavity is used to compute a footprint, and the appropriate entry in the hash table is accessed. For each pair, *(model, concavity)*, appearing in the hash table at that location, we compute the appropriate affine transformation to the associated model, and attempt to verify an instance of the model in the image.

If the concavity entrances are distinctive enough, the complexity of the recognition stage will be this time linearly dependent on the number of concavities in the scene and the number of scene vertex points, namely, $O(k \times n)$.

### 2.9.2. Convex Curve Matching

In case we have a model with a convex boundary curve, or if we wish to recognize non-convex models with all concavity entrances occluded, a different method is

needed. (It is interesting to point out that for these cases, the point classification method of [43] is also unapplicable.) We must resort to a strategy with a greater time complexity, since there is no 'natural' affine base, as the one defined by a concavity. Specifically, to construct the hash table for a given convex model, we iterate over all pairs of boundary points. For each pair, we join the two points with a line, which will be called the 'base line.' There are two most distant curve points from the base line, one on each side of the base line. (If this point on one side is not defined uniquely, take the leftmost such point). Each most distant point together with the endpoints of the base line form an affine basis triplet. To each such basis triplet corresponds a convex region bounded by the base line and the convex body, containing all three basis points. This is the 'basis region' (see Fig. 3).


**Figure 3.** A basis triplet of a convex object.

As before, we can use a *footprint* based on the normalized basis region to create an entry to a hash table. In this case, for each pair of boundary points, we have a separate footprint. Thus the hash table entries contain the identification of the model and the identification of the basis triplet. For recognition, we judiciously choose a pair of points in the scene on the boundary of a convex curve, find an associated convex 'basis region,' and compute the footprint. For this footprint (properly quantized), we check the appropriate entry in the hash table, and extract the pairs *(model, basis triplet)* appearing there. For each such relevant model with the appropriate basis triplet, we compute the corresponding affine transformation between the model and the scene, and verify their correspondence.

Observe that convex bodies usually intersect at concave angles (in [24] they are called breakpoints). Thus for the recognition step, it will be enough to examine only one pair of points (one base line) for each convex 'protrusion', delimited by two consecutive breakpoints. Hence, if a scene has $k$ convex boundary subcurves (delimited by consecutive breakpoints), the recognition stage of the algorithm wil be of the order $k \times n$ , where $n$ is the number of objec: vertices. That is, the recognition phase will be very efficient.

### 2.9.3. 3-D objects

The methods described in this paper naturally extend to the recognition of 3-D objects. In the case where accurate 3-D depth data is available, an entirely similar procedure can be developed based on extraction and matching of points in 3-D, subject to a rigid transformation. Here, we discuss the more general problem of recognition of 3-D objects from a single 2-D view. We assume that the variation in depth of the objects is small compared to the depths of the object centroids, so that the perspective projection is well-approximated by an affine transformation.

We distinguish three approaches to the problem. The appropriate method will depend upon the complexity of the models, and the robustness of the feature extraction.

If the objects can be approximated by polyhedral solids, then we may build a database of 2-D models representing the 'almost' planar faces of each 3-D model. The problem then reduces to recognition of these flat surfaces, according to the methods of the previous sections. The faces may be partly obscured by the presence of other 3-D objects in the line-of-sight to the object. Complete identification of the object and its orientation can be verified by the consistent identification of other faces of the 3-D object in the appropriate locations [34].

Alternatively, we may discretize the space of viewing directions, and produce a nearly flat model of each 3-D object from each given viewing direction. Recognition proceeds by identifying objects and the viewing direction among the database of models, which may have been subjected to a similarity transformation. In this approach, there will be many models, but due to the fact a similarity transformation is sought rather than an affine transformation (Section 2.7), there will be a reduced complexity in the recognition phase.

Finally, in the same way that a triplet of points in the 2-D case can be used as an affine basis, four points on the surface of a 3-D model, providing they are non-coplanar, define a 3-D basis. Suppose that we choose four points in the scene, and posit a match with four corresponding points in a model. The correspondence defines a 3-D affine transformation of the object. The match can be verified quickly, by checking whether other points of the model appear in the scene according to the affine transformation. The checking can be made faster by a hashing scheme (although, one that is different than the hashing method presented earlier). However, methods to speed the search for an appropriate set of four points in the scene can contribute greatly to the efficiency of this approach.

## 3. Experimental Results

We have implemented the point matching, non-convex curve matching and best least-squares matching algorithms.

In the first set of figures we show recognition of industrial parts (pliers) in composite scenes. Here the point matching algorithm and least squares matching algorithm were applied. Figs. 4a and 4b are the original gray scale images of two models (pliers), and Fig. 4c and 4d show the extracted 'interest points' of the models, which are points of sharp concavities and convexities. In Fig. 6a we see an image of the pair of pliers of Fig. 4a rotated, translated and tilted at about 40 degrees (observe the different lengths of both handles in the image). The recognition algorithm was performed to obtain a number of matching basis-triplets. The corresponding affine transformations were calculated and for each such transformation the transformed model was superimposed on the scene of Fig. 6a. Fig. 6c shows such a transformation computed according to a basis triplet which gives a somewhat noisy match. This solution is significantly improved by the best least-squares match which is given in Fig. 6d, and was calculated using all the points which were

recognized as model points by the basis triplet of Fig. 6c.

In Fig. 5 we see an image of a composite overlapping scene of the two pliers, the extracted 'interest points', and the recognition results. Note that in Fig. 5b we have additional 'interest points' that do not correspond to 'interest points' in the original models, but are created by the superposition of the two objects. Also, one can see that a number of the original 'interest points' are occluded in the scene.

The second set of figures deals with recognition of some household items. Fig. 7a is the original gray scale image of a pizza cutter. In Fig. 7b the concavity entrances are marked by the dashed lines, and the concavity basis triplets are displayed. Fig. 8a is a composite scene of the pizza cutter and a spatula. The image was taken by a significantly tilted camera resulting in an affine distortion of the model. In Fig. 8b the concavity entrances of the composite scene are marked, and the basis triplet points are displayed. The algorithm of Section 2.9.2 was applied to this scene, resulting in the recognition of the pizza cutter displayed in Fig. 8c.

## Acknowledgements

Work described in this paper was supported by Office of Naval Research Grants N00014-82-K-0381 and N00014-85-K-0077 Work Unit NR-4007006, and National Science Foundation Grants NSF DCR-83-20085 and IRI-8703335.

## References

[1] Hummel, Robert A., B. Kimia, and S. Zucker, "Deblurring gaussian blur," *Computer Vision, Graphics, and Image Processing* **38**, pp. 66-80 (1987).

[2] Hummel, Robert and Robert Moniot, "Solving ill-conditioned problems by minimizing equation error," *Proceedings of the IEEE First Insternational Conference on Computer Vision*, pp. 527-533 (1987).

[3] Hummel, Robert and Robert Moniot, "A network approach to reconstructions from zero-crossings," *Proceedings of the IEEE Computer Society Workshop on Computer Vision*, pp. 8-13 (November, 1987).

[4] Hummel, Robert, "The scale-space formulation of pyramid data structures," pp. 107-123 in *Parallel Computer Vision*, ed. Len Uhr, Academic Press, New York (1987).

[5] Hummel, Robert, "Representations based on zero-crossings in scale-space," *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pp. 204-209 (June, 1986).

[6] Landy, M., "A parallel model of the kinetic depth effect using local computations," *Journal of the Optical Society of America (A)* **4**, pp. 864-876. (1987).

[7] Hummel, Robert A. and Steven W. Zucker, "On the foundations of relaxation labeling processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-5**, pp. 267-287 (May, 1983).

[8] Landy, Michael S. and Robert A. Hummel, "A brief survey of knowledge aggregation methods," *Proceedings of the International Conference on Pattern Recognition*, pp. 248-252 (October, 1986).

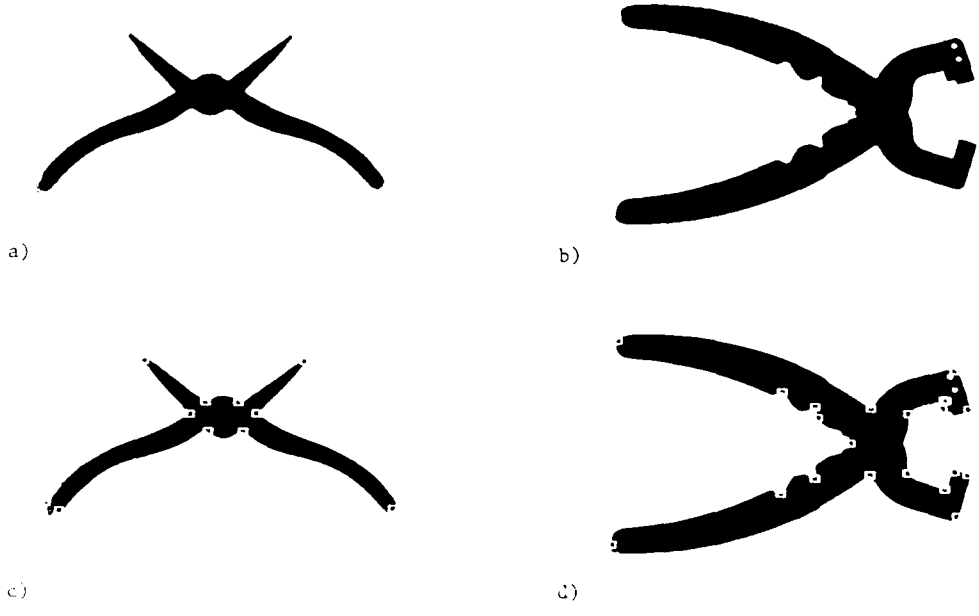



Figure 4 : The models of the two pliers and their extracted 'interest points'.

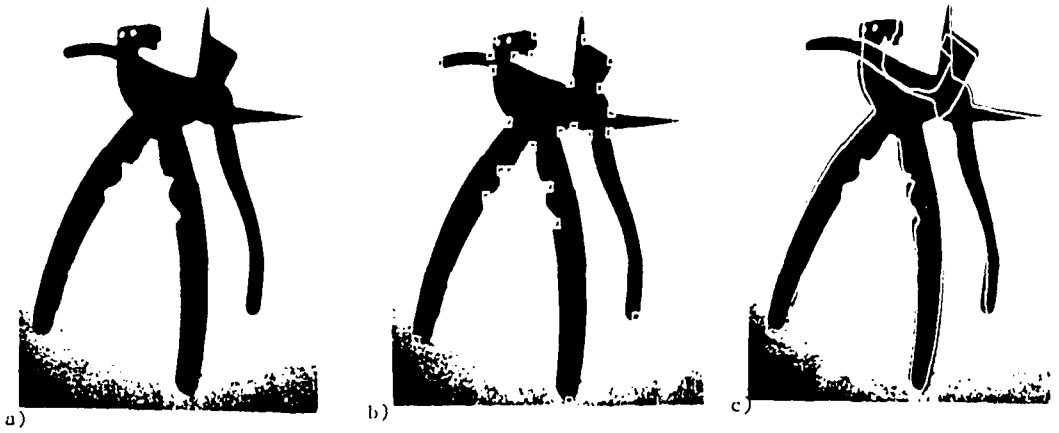



Figure 5 : a) A composite scene of the pliers of Fig.1 (observe different lengths of handles due to the tilt). b) Extracted 'interest points'. c) Recognition of the models in the scene.

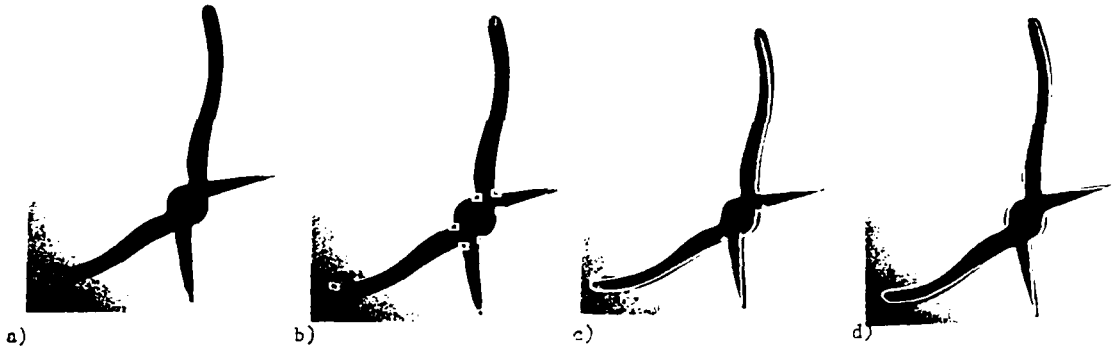



Figure 6 : a) The plier rotated and tilted in space (see different length of handles). b) Extracted 'interest points'. c) Matching based on one 'basis triplet' correspondence. d) Best least squares affine correspondence.

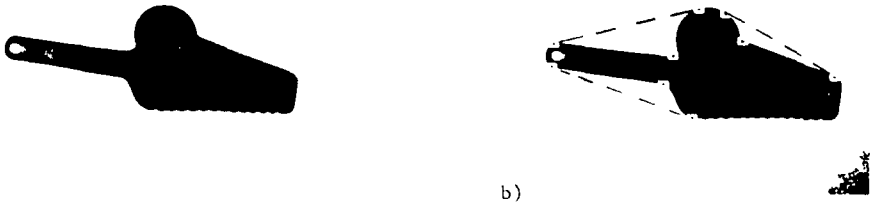



Figure 7 : a) gray scale image of a pizza cutter. b) concavity entrances and basis triplets.

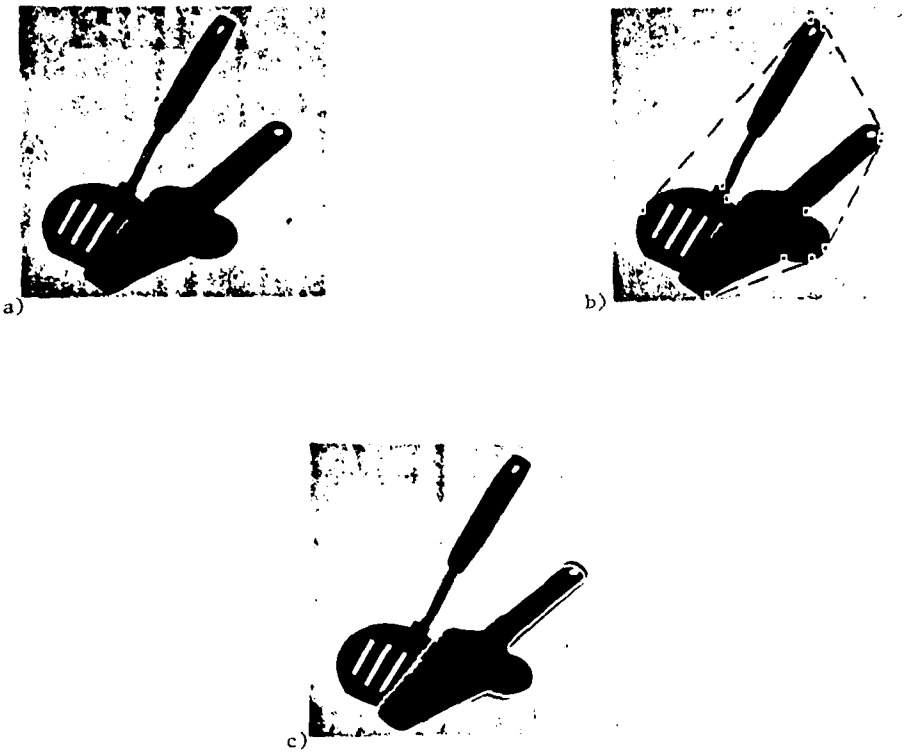



Figure 8 : a) a composite scene. b) concavity entrances and basis triplets of the scene. c) recognition of the pizza-cutter.


[9] Hummel, Robert A. and Michael S. Landy, *A statistical viewpoint on the theory of evidence*, To appear: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. March, 1988.

[10] Hummel, Robert and M. Landy, "Evidence as opinions of experts," *Proceedings of the "Uncertainty in AI" Workshop*, pp. 135-143 (August 8-10, 1986).

[11] Wolfson, Estarose and Eric L. Schwartz, "Computing minimal distances on polyhedral surfaces ," *IEEE Pattern Analysis and Machine Intelligence*, (1988). In press.

[12] Schwartz, Eric L., Alan Shaw, and Estarose Wolfson, "A numerical solution to the generalized map makers problem," *IEEE Pattern Analysis and Machine Intelligence*, (1988). In press.

[13] Yeshurun, Yehezkel and Eric L. Schwartz, "Cepstral filtering on a columnar image architecture: a fast algorithm for binocular stereo segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence* , (1988). In press.

[14] Yeshurun, Yehezkel and Eric Schwartz, "Shape description with a space variant sensor: algorithms for scan-path,fusion and convergence over multiple scans," *IEEE Pattern Analysis and Machine Intelligence* , (1988). In press.

[15] Bastuscheck, M. and J. Schwartz, "Experimental implementation of a ratio image depth sensor," pp. p. 1-12 in *Techniques for 3-D Machine Perception*, ed. A. Rosenfeld, Elsevier (North-Holland) (1986).

[16] Carrihill, Brian and Robert A. Hummel, "Experiments with the intensity ratio depth sensors," *Computer Vision, Graphics, and Image Processing* **32**, pp. 337-358 (1985).

[17] Menczel, Y., "Description of shape using orientation and propogation flow," Ph.D. Thesis, Dept. of Computer Science, NYU (Oct., 1986).

[18] Schwartz, J.T. and M. Sharir, "Identification of Partially Obscured Objects in Two Dimensions by Matching of Noisy 'Characteristic Curves'," *The Int. J. of Robotics Research* **6** (2), pp. 29-44 (1987).

[19] Bastuscheck, C. M., E. Schonberg, J.T. Schwartz, and M. Sharir, "Object Recognition by 3-Dimensional Curve Matchin," *Int. J. of Intelligent Systems* **1**(2), (1986).

[20] Wolfson, H., "On Curve Matching," *Proceedings of the IEEE Computer Society Workshop on Computer Vision*, pp. 307-310 (November, 1987).

[21] Schwartz, J. and H. Wolfson, "Improved shape-signature and matching methods for model-based robotic vision," *Proceedings of the Workshop on Space Telerobotics* **II**, pp. 103-109 (Jan., 1987). NASA, JPL

[22] Kishon, E., "Simultaneous matching of curves in three dimensions," NYU Computer Science TR 263 (Nov., 1986).

[23] Wolfson, H., E. Schonberg, A. Kalvin, and Y. Lamdan, "Solving Jigsaw Puzzles Using Computer Vision," *Annals of Operations Research*, (1987). In Press.

[24] Kalvin, A., E. Schonberg, J.T. Schwartz, and M. Sharir, "Two Dimensional Model Based Boundary Matching Using Footprints," *The Int. J. of Robotics Research* 5(4), pp. 38-55 (1986).

[25] Hong, J. and H. J. Wolfson, "An Improved Model-Based Matching Method Using Footprints," *Robotics Report 137*, Computer Science Div., Courant Inst. of Math., NYU, (1987).

[26] Kishon, E. and H. Wolfson, "3-D Curve Matching," *Proceedings of the AAAI Workshop on Spatial Reasoning and Multisensor Fusion*, pp. 250-261 (October, 1987).

[27] Lamdan, Y., J.T. Schwartz, and H. J. Wolfson, "Object Recognition by Affine Invariant Matching," *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, (June, 1988). To appear.

[28] Hong, J. and X. Tan, "The similarity between shapes under affine transformation," NYU Robotics Research Report 133 (Dec., 1987). Submitted.

[29] Hummel, Robert A., "Connected component labeling in image processing with MIMD architectures," in *Intermediate-level Image Processing*, ed. M. J. B. Duff, Academic Press, Bonas, France (1986).

[30] Hummel, Robert and Kaizhong Zhang, "Dynamic processor allocation for parallel algorithms in image processing," *Proceedings of the Optical and Digital Pattern Recognition session of the SPIE Conference on EO-Imaging, SPIE Vol. 754*, pp. 268-275 (January, 1987).

[31] Lowe, D.G., *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers (1985).

[32] Goldberg, R. and D. Lowe, "Verification of 3-D parametric models in 2-D image data," *Proceedings of the IEEE Computer Society Workshop on Computer Vision*, pp. p. 255-257 (Nov., 1987).

[33] Hummel, Robert and David Lowe, "Computing large-kernel convolutions of images," NYU Robotics Technical Report 84 (1986).

[34] Lowe, D.G., "The Viewpoint Consistency Constraint," *Int. J. of Computer Vision* 1 (1), pp. 57-72 (1987).

[35] Lowe, D., "Three dimensional object recognition from single two-dimensional images," *Artificial Intelligence* 31, pp. 355-395 (1987).

[36] Lamdan, Y., J. T. Schwartz, and H. Wolfson, "On recognition of 3-D objects from 2-D images," *Proceedings of the IEEE Int. Conference on Robotics and Automation*, (April, 1988). To appear.

[37] Chin, R.T. and C.R. Dyer, "Model-Based Recognition in Robot Vision," *ACM Computing Surveys* 18 (1), pp. 67-108 (1986).

[38] Ayache, N. and O.D. Faugeras, "HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects," *IEEE TPAMI* 8 (1), pp. 44-54 (Jan., 1986).

[39] Bolles, R.C. and R.A. Cain, "Recognizing and Locating Partially Visible Objects: The Local Feature Method," *The Int. J. of Robotics Research* 1 (3), pp. 57-82 (1982).

[40] Turney, J.L., T.N. Mudge, and R.A. Volz, "Recognizing Partially Occluded Parts," *IEEE TPAMI* 7 (4), pp. 410-421 (July 1985).

[41] Grimson, W.E.L. and T. Lozano-Perez, "Model Based Recognition and Localization from Sparse Range or Tactile Data," *The Int. J. of Robotics Research* 3 (3), pp. 3-35 (1984).

[42] Besl, P.J. and R.C. Jain, "Three-Dimensional Object Recognition," *ACM Computing Surveys* 17 (1), pp. 75-154 (1985).

[43] Huttenlocher, D.P. and S. Ullman, "Object Recognition Using Alignment," *Proc. of the 1'st Int. Conf. on Computer Vision*, pp. 102-111 (1987).

[44] Thompson, D.W. and J.L. Mundy, "Three-Dimensional Model Matching from an Unconstrained Viewpoint," *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 208-220 (1987).

[45] Klein, F., *Elementary Mathematics from an advanced standpoint ; Geometry*, Macmillan, New York (1925 (Third edition)). (English translation)

[46] Moravec, H., "Towards automatic visual obstacle avoidance," *Proceedings of the 5th IJCAI*, p. p. 584 (1977).

[47] Barnard, S. and W. Thompson, "Disparity analysis of images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, p. 333 (1980).

[48] Yaglom, I.M. and V.G. Ashkinuze, *Ideas and Methods of Affine and Projective Geometry*, (in Russian) 1962.

[49] Cyganski, R. and J.A. Orr, "Applications of Tensor Theory to Object Recognition and Orientation Determination," *IEEE PAMI* 7 (6), pp. 662-673 (Nov. 1985).

[50] Cyganski, R., J. Orr, T. Cott, and R. Dodson, "Development, implementation, testing, and application of an affine transform invariant curvature function," *Proceedings of the 1st Int. Conf. on Computer Vision*, pp. 496-500 (1987).

[51] Schwartz, J.T. and M. Sharir, "Some Remarks on Robot Vision," *Trans. of 3'rd Army Conf. on Applied Math. and Computing*, pp. 1-36 (May 1985).

[52] Mehlhorn, K., *Multi-dimensional Searching and Computational Geometry*, Springer (1984).

# Constructing Simple Stable Descriptions for Image Partioning *

Yvan G. Leclerc

Artificial Intelligence Center, SRI International
333 Ravenswood Avenue, Menlo Park, Ca 94025

## Abstract

This paper presents a new formulation of the image partitioning problem: construct a complete and stable description of an image, in terms of a specified descriptive language, that is simplest in the sense of shortest description length. A suitable descriptive language that results in intuitively satisfying partitionings can be limited to a low-order polynomial description of the intensity variation within each region and a measure of the length of the region boundaries; other aspects of the shape and semantic properties of the regions need not be explicitly considered. Experiments performed on a variety of both natural and synthetic images demonstrate the superior performance of this approach over partitioning techniques based on clustering vectors of local image attributes and over standard edge-detection techniques.

## 1 Introduction

The partitioning problem is one of the most important unsolved problems in computer vision. While impossible to define precisely in its broadest sense, the partitioning problem requires a procedure to delineate regions in an image that correspond to semantic entities in the scene (i.e., objects, processes) or coherent regions or structures in the image. In practice, most partitioning techniques are designed to identity regions that are homogeneous in some set of local image attributes, such as intensity, color, and texture, or to detect the boundaries between regions based on discontinuities of these local image attributes.

In this paper, we do not critically distinguish between the partitioning problem and subsequent steps of the scene analysis process. We formulate this analysis as one of finding a "best" description of the image in terms of some specified descriptive language; partitioning differs from later steps only in the simplicity of the vocabulary employed. The principal components of any solution thus include the specification of the descriptive language and a computationally feasible procedure for selecting a best description.

We will show that a suitable descriptive language for providing an intuitively satisfying solution to the partitioning problem for a broad variety of images can be limited to a low-order polynomial description of the intensity variation within each region and a measure of the length of the region boundaries; other aspects of the shape and semantic properties of the regions need not be explicitly considered.

Our principal contribution in this paper is in providing a set of criteria, and a corresponding computational definition, for obtaining a best description of the image. We require that the description be both stable (minor perturbations in the viewing conditions should not alter the description) and complete in the sense that all the intensity variation, up to uncorrelated white noise, is explained by the polynomial intensity model. That is, our description will differ from the actual image only to the extent that the residuals can be characterized as independent samples from a normal distribution (whose parameters are not known *a priori* and may differ from region to region). Finally, we select the simplest (in the sense of shortest description length) description which satisfies our previous requirements. We describe a continuation-method-based computational procedure to construct the desired optimal description.

Experiments performed on a variety of both natural and synthetic images demonstrate the superior performance of the specified approach over partitioning techniques based on clustering vectors of local image attributes. Examples are included to show that naturally occurring subjective edges, invisible to local techniques, are correctly placed, and that the "ugly" mistakes typical of local techniques are largely avoided.

In the following section, I describe the general framework for formulating scene analysis probems. This framework is the construction of the best description of an image in an appropriate descriptive language, where we define criteria for descriptive languages and the definition of best. I then define the concept of a minimal-length information-preserving description (a formal definition of what we mean by a simplest complete description), show how to design optimal languages for such descriptions, and show that these descriptions, followed by simple tests for stability, satisfy the criteria just mentioned. Then, I develop a specific descriptive language for image partitioning, derive an algorithm for finding the simplest description in this language, and illustrate the application of the language and algorithm to several images.

## 2 General Framework

The general framework of this approach can be described intuitively as constructing the best description of an image in some specified descriptive language. The choice of descriptive language and what is meant by "best" is, of course, strongly task dependent. However, not all choices are reasonable, and I argue that the following set of criteria are important, and perhaps even necessary, constraints on the choice of language and what is meant by best. These criteria will be used to formulate the problem of finding the best description as that of finding the simplest description, as described in the next section.

The first criterion is a constraint on the descriptive language

*Support for this work was provided by the Defense Advanced Research Projects Agency under contract MDA903-86-C-0084.

alone, namely that the descriptive language must be *complete*. That is, all descriptions in the language must exactly determine a single image. Thus, what one usually describes as "noise" must be included as part of the descriptive language. Note that completeness means that a given description yields only one image, but there may be many, perhaps even uncountably many, different descriptions for a given image.

The second criterion is a constraint on both the language and the definition of best, namely *computational feasibility*; that is, the best description of an image (or at least something very close to it) must be constructable in a reasonable amount of time.

Of crucial importance to any system that purports to find the best description of an image is the ability to determine when the image (or, more generally, some portion of the image) lies outside the range of the descriptive language. This leads to further criteria, which must be satisfied for all (or at least a very large fraction) of the images for which the language is appropriate. Thus, failure to satisfy either of these further criteria is a strong indication that the language is inappropriate.

The third criterion, then, is that the best description of an image must be an *efficient* description. A weak form of this criterion is that the best description must be shorter in length than the image itself, as suggested in Georgeff and Wallace [6]. A stronger form, which can only be defined roughly here, is that the complexity of the description should not exceed the complexity one would expect for the given image.

The fourth and final criterion is that the best description of an image must be a *stable* description. Of course, since descriptions are complete, *any* change in the image causes *some* change in the description. Thus, stability cannot be defined in the obvious manner. Instead, we define stability in the more general sense that *some portion* of the best description is invariant to *certain classes* of image changes. A further consequence of the completeness of descriptions is that the class of image changes just mentioned are necessarily reflected in changes to other portions of the description. Thus, the stability of one portion of the best description is a function of changes to other portions of the best description.

As motivation for these criteria, consider the following example, where a complete three-dimensional description of a scene, including a complete camera model, has been computed from a single image. Clearly, we should expect the volumetric (three-dimensional shape) portion of the description to almost always remain the same, given a new image of the same scene differing only slightly in, say, lighting, surface coloration, or viewpoint. In other words, we should expect the volumetric portion of the description to remain invariant to (i.e., be stable with respect to) the class of image changes corresponding to slight changes in lighting, surface coloration, and/or viewpoint, but not to image changes corresponding to changes in the shapes of objects, for example.[1]

Of course, when supplied with a single image, we cannot directly test for stability by analyzing another image of the scene with slightly changed characteristics. (Indeed, for changes like surface coloration, this is not feasible even if we had the opportunity to take a new picture!) Instead, we are left with a slightly weaker, but still crucial expectation: that the volumetric portion of the description should remain the same given a synthetically generated image derived from a new description—one where the portions of the new description relating to, say, lighting, surface coloration,and/or viewpoint are slightly different from the first description. This expectation is precisely an example of the stability criterion defined above. Furthermore, we now see the motivation for the completeness criterion: without it, we could not have generated a unique new image from the modified description.

Now, it is clearly infeasible to test directly for stability in the above fashion because that would entail generating and analyzing a large number of synthetic images. Instead, one must demand that the language and the algorithm that computes the best description be designed in such a way that, together, they guarantee stability in this sense for the class of images for which the language is designed. Or, at least allow for a computationally inexpensive determination of some measure of stability of the description (perhaps on a part by part basis).

In the next section, I introduce the notion of a minimal-length information-preserving description, and show that one can design descriptive languages such that, for almost all images for which the language was designed, the simplest complete description is also a stable description. Such descriptions are called minimal-length information-preserving descriptions of an image in terms of a given descriptive language. This leads us to formulate the problem of finding the best complete and stable description as that of finding the minimal-length description, followed by simple tests for stability.

## 3 Minimal-Length Information-Preserving Descriptions

The idea that simpler descriptions are better than more complex ones is a strongly intuitive idea that goes back, at the very least, to the ancient Greeks. It embodies not only the notion that simpler descriptions are better because they are easier to use in many ways, but also the body of scientific (and personal) experience that tells us there is almost always a simpler description of a set of observations than their mere tabulation.

However, the idea that simpler is better is quite vague: What exactly does it mean for one description to be simpler than another? One possible answer is that the number of degrees of freedom, or distinct and independent variables in the description, should be the measure of simplicity. Take, for example, the classical curve-fitting problem, where one is presented with an ordered set of numerical observations that can purportedly be described as points along some mathematically defined curve. The simplest description, then, should be the one that requires the fewest number of parameters to define the curve. But, even for such a simple problem, one immediately sees that the definition as stated is still somewhat vague.

First, the number of parameters required to define a curve depends very much on the vocabulary of curves one brings to bear. For example, if the observations were actually equally spaced points on a quadratic curve, but one attempted to describe them using sinusoids (as in a discrete Fourier transform), one would require as many parameters as there are observations; however, a polynomial representation would require only six parameters (three specifying the number of observations, spacing, and order of the polynomial, and three specifying the curve), independent of the number of observations. Thus, one would be

[1] In effect, Binford [1] calls stability with respect to change in viewpoint the assumption of general position. In this sense, general position is a special case of our notion of stability.

inclined to say that the polynomial description is the simpler of the two for these observations. However, if one is allowed to use any possible mathematical curve, one must also specify which of the infinite class of curves the parameters refer to (polynomials vs. sinusoids vs. ...). Since this clearly requires an infinite number of parameters, one is left with the inescapable conclusion that the vocabulary of curves (or, more generally, the language in which the description is expressed) must be restricted in some sense, or else one will always need more parameters than observations.

A second fundamental problem with this definition of simplicity is that almost all phenomena, and hence observations of them, inherently have a stochastic component. At the very least, the observations will be corrupted in some stochastic manner, even if the underlying phenomenon is purely deterministic. Thus, for our curve-fitting example, even if one could specify the underlying curve using a few variables, one would still need to describe the point by point deviations from the curve (either directly or in some appropriate parameter space) in order to have a complete description, and this would require at least as many variables as observations! Again, one is left with more variables than observations.

The information-theoretic answer to this quandary is to reduce the idea of an independent variable to its simplest form: a bit. The measure of simplicity then becomes the number of bits in the description that some computationally effective procedure requires in order to reproduce the observations. That is, the simplest description will be the one that requires the fewest number of bits for which the procedure reproduces the input. This, of course, demands the prior specification of the computationally effective procedure, thus specifying the language in which the description is expressed. In other words, the notion of simplicity in this formalism becomes a relative one, one which strongly depends on the choice of descriptive language.

The advantage of this formulation is that one can define procedures for reproducing both deterministic and stochastic processes. Moreover, there are provably optimal procedures for reproducing stochastic processes (optimal in the sense of requiring the fewest number of bits of description per bit of input, on average). When using such optimal procedures, there is a natural trade-off between the allocation of bits for the deterministic versus the stochastic processes. For the curve-fitting problem, this amounts to a natural trade-off between the number of variables required to specify the curve and the deviations versus the number of bits per variable. This is described in detail in the following subsection. For an excellent introduction to the formal theory of minimal-length descriptions, see Rissanen's discussion [9].

Before we enter the next subsection, a more formal definition of the minimal-length description problem, as we will use it, is required. We define the problem in terms of an *encoder*, whose input is the set of observations in the form of a bit string, $\mathbf{y}$, and whose output is supplied to a *decoder*, which uses this to produce the *output*.

When the output of the decoder is identical to the input $\mathbf{y}$, then the output of the encoder is called an information-preserving description of the input, denoted $\mathcal{D}_i^j(\mathbf{y})$, written in the *descriptive language* $L_i$ of the decoder. The superscript $j$ indicates that there may, in general, be more than one possible description for a given $\mathbf{y}$. We shall call one minus the ratio of the number of bits in the description to the number of bits in the input the *efficiency* of the description. By definition, information-preserving descriptions satisfy the completeness criterion of the previous section.

The most efficient description, $\mathcal{D}_i^*(\mathbf{y})$, is the minimal-length information-preserving description of the input written in the descriptive language of the decoder. We will say that a descriptive language is *appropriate* for a class of input strings when it yields a unique minimal-length description for almost all strings in the class.

### 3.1 Minimal-Length Descriptions of Ergodic Processes

Consider the case when the input string is generated by a known ergodic process $\mathcal{F}$, that is,

$$\mathbf{y} = \mathbf{F}(\mathbf{x}),$$

where $\mathbf{F}$ is a deterministic, ergodic, function and $\mathbf{x} = \{x_0, x_1, \ldots, x_{n-1}\}$ is drawn from a known unchanging distribution. (The length of the vector, $n$, may also be a random variable.) The probability of the input, denoted $P(\mathbf{y})$, is therefore defined for every input string, and the pair $(\mathbf{F}, \mathbf{x})$ constitutes a description of the input. When $\mathbf{F}$ is not uniquely invertible, i.e., when a given input string can be produced by many different $\mathbf{x}$s, the problem arises of how to choose a single $\mathbf{x}$ for a given input.

One way of choosing a single vector, out of all the possible vectors that can produce the input, is to choose the most likely one, $\mathbf{x}^*$. This is called the maximum *a posteriori*, or MAP, strategy because one maximizes the *a posteriori* likelihood of $\mathbf{x}$, $P(\mathbf{x} \mid \mathbf{y})$, which is a function of the prior probabilities of $\mathbf{x}$ and the function $\mathbf{F}$.

Unfortunately, for many functions and distributions of interest, there may be many equally likely vectors that produce a given input string. For example, if $\mathbf{F}(\mathbf{x}) = x_0 + x_1$, and $x_0$ and $x_1$ are each drawn from the same uniform distribution, then there will generally be many most-likely pairs for a given input. If this multiplicity occurs for a significant fraction of the input strings, then the problem of choosing a single most likely vector is ill-defined. In the terminology of the previous section, the most-likely description would be unstable for that fraction of input strings.

The minimal length description solution to the problem of choosing a single description is based on the observation that it is always possible to design an optimal descriptive language $L_{\mathcal{F}}$ for an ergodic process $\mathcal{F}$ such that the shortest description of the input has length

$$|\mathcal{D}_{\mathcal{F}}^*(\mathbf{y})| = -\log_2 P(\mathbf{y}) \qquad (1)$$

bits[2] [9]. (We will see precisely how to do this for certain kinds of ergodic processes later.) Such a descriptive language is optimal in the sense that no other descriptive language can expect to produce a shorter description than this, on average.[3] A consequence

[2] For some distributions, one would need to encode an infinitely long input string in order to achieve exactly this efficiency. A more precise statement is that we can achieve an efficiency as close to this optimum as we like by encoding sufficiently large chunks of the input string at a time.

[3] This is not to say that no other descriptive language can do better on any given finite input string, but only that no other language can do better on average. Or, equivalently, no other language can do better for arbitrarily long input strings.

of this optimality is that there exists a unique shortest description for every input string,[4] because otherwise there would exist "wasted" descriptions (the ones that map to the same input) that *could* have been used for other inputs but were not; hence one *could* have devised a more efficient descriptive language that made use of these wasted descriptions. Note, however, that there are always many different optimal descriptive languages for a given ergodic process, but they are equivalent to each other in the sense that there exist one-to-one mappings between them (as a consequence of the uniqueness of description just mentioned).

When there is a unique most-likely vector for each input string, the minimal-length description solution is equivalent to the MAP solution in the sense that there is then a one-to-one mapping between each shortest description and $\mathbf{x}^*$ (again, as a consequence of the uniqueness of description).

When there is not a unique most-likely vector, one can view the optimal descriptive language as a way of describing a different function and distribution of vectors (with fewer elements, or degrees of freedom) that produce the same distribution of input strings, but for which there is a unique most-likely vector for each input string. Intuitively, the optimal descriptive language merges together hidden or unobservable variables (such as $x_0$ and $x_1$ in the example above) and replaces them by observable variables (such as $x = x_0 + x_1$).

In summary, one can define optimal descriptive languages when the distribution of input strings is known *a priori*. When the input is generated by a known ergodic process that has a unique most-likely vector for each input string, the shortest description in the optimal language is equivalent to the most-likely vector. Otherwise, the shortest description is equivalent to the most-likely vector of a different ergodic process that produces the same distribution of input strings, but that does have a unique most-likely vector.

### 3.2 Minimal-Length Descriptions for Specified Descriptive Languages

Now, one can turn the argument around and ask whether it is reasonable to simply define a descriptive language and then compute the shortest description for each input string. Certainly we can do so by assuming that the input is an ergodic process with a unique most-likely vector, and design the language accordingly. We know that, *if the input follows the distribution that the language was designed for*, the shortest description will then be the most-likely vector. Furthermore, because optimal descriptive languages have unique shortest descriptions for all inputs, any descriptive language that yields a unique shortest description for all input strings is an optimal descriptive language for some ergodic process. Therefore, *if we wish to interpret the shortest description as the description of the most-likely vector*, then choosing a descriptive language is equivalent to assuming that the input is generated by that ergodic process.

However, as we saw above, there are many different optimal languages for a given distribution of input strings. Thus, additional criteria, such as those mentioned in the previous section, must be brought to bear in the design process. In particular, one must have a way of deciding when the descriptive language is inappropriate for the input, or equivalently, when the input does not follow the distribution that the language was designed for. Unfortunately, with the exception of inputs that simply cannot be described in the language (because the probability of that input was assumed to be zero), there are no criteria that we can apply with absolute certainty to finite-length input strings. Even the criterion that the length of the description must be shorter than the length of the input string is not an absolute certainty—we may just have been very unlucky for that particular draw!

Instead, when dealing with ergodic processes, we must be satisfied with somewhat weaker, probabilistic criteria. In particular, if we design the language so that the criteria of the previous section are almost always met when the input follows the appropriate distribution, then failure to satisfy these criteria for a given input string is a strong, but not absolutely certain, indication that the language is indeed inappropriate for that input.

To make all of this concrete, let me illustrate the design of some optimal descriptive languages, using the criteria of the previous section, with three examples. The first two examples illustrate the design of optimal descriptive languages that are one-to-one mappings of the input, i.e., for which there is exactly one description for a given input. Since the mappings are one-to-one, the stability criterion is inapplicable, and we are left with only the efficiency criterion.

The third example builds on the first example for inputs that result from complex combinations of several independent processes, in much the same way that images are the results of several independent processes. In this case there are many possible descriptions of the input, and the encoder must, in some manner, determine the shortest of these descriptions. We will describe a specific optimization algorithm for this language that generalizes to the more complex languages required for image partitioning. We will show how to determine the stability of the shortest description, and how to use this measure of stability to determine the appropriateness of the descriptive language for the given input. This final example is the basis for the image partitioning problem in the remainder of the paper.

### 3.3 Example 1: Independent Symbols

The purpose of this example is to illustrate the design of an optimal descriptive language for input strings consisting of symbols independently drawn from a known distribution. (This kind of language is useful for the noise component of images, as we shall see in the third example.) In this example, $\mathbf{F}$ is the identity function, and $\mathbf{y} = \mathbf{F}(\mathbf{x}) = \mathbf{x} = \{x_0, x_1, \ldots, x_{n-1}\}$, where $x_i$ is one of the three symbols '00,' '01,' '10,' independently drawn with probability 0.5, 0.25, and 0.25, respectively.

From Eq. (1) above, we can design an optimal descriptive language $L_1$ such that the description length is

$$\begin{aligned} |\mathcal{D}_1^*(\mathbf{y})| &= -\log_2 P(\{x_0, x_1, \ldots, x_n\}) \\ &= -\log_2 \prod_{i=0}^{n-1} P(x_i) \\ &= \sum_{i=0}^{n-1} -\log_2 P(x_i). \end{aligned}$$

Since $-\log_2 P(\text{'00'}) = 1$ and $-\log_2 P(\text{'01'}) = -\log_2 P(\text{'10'}) = 2$, a code devoting exactly one bit for '00' and two bits each for '01' and '10,' such as the *prefix* code

'00' — '0'

[4] The phrase "for every input string" used here and elsewhere is short for "for every input string that has nonzero probability of occurrence."

'01' → '01'
'10' → '10'

is an optimal code. Thus, on average, the optimal encoding of an input string containing $n$ symbols (or $2n$ bits) requires only $1.5n$ bits. The efficiency of the optimal encoding is therefore $1 - 1.5/2 = 0.25$.

In general, $-\log_2 P(x_i)$ is not an integer, so we might encode several symbols at once (called block encoding) in order to achieve something close to the optimal encoding length, but the principle remains the same.

With the exception of input strings containing the symbol '11,' (which is assumed to be impossible for this example), there is no way of determining with certainty that a given finite input string has not been drawn from the distribution we've assumed. Furthermore, the weak form of the efficiency criterion (that the length of the description be shorter than or equal to the length of the input) is useless because it is trivially satisfied for any input string not containing the symbol '11.' However, it is possible to define the stronger form of the efficiency criterion for this example, namely that the description should not be too complex relative to the input. Specifically, we can calculate the range within which the ratio of bits of description per bit of input should lie, say, 99.9% of the time, as a function of $n$, and reject descriptions that fall outside of this range.

To reiterate what was said in the initial subsection, the more general way of determining that the descriptive language is inappropriate is to see whether a different descriptive language could encode the input in fewer bits. But, to make the comparison, one must have a common language in which to write these different descriptive languages, and the length of the description of the descriptive language must be taken into account! In other words, one cannot get away from the dependency on a base descriptive language of some kind. The efficiency criterion in the previous paragraph is merely a computationally inexpensive approximation to this more general method.

Although the first-order statistics of an input string (the probability of occurence of a symbol) capture some aspect of the structure of the input (and indeed, capture it entirely when the symbols are independent and identically distributed as in this example), the next two examples illustrate that much more is needed in general.

### 3.4 Example 2: Correlated Symbols

Consider a similar case to the one above where the three symbols '00,' '01,' and '10' occur with the same probabilities as before, but such that they occur only as one of two sequences, say '00000110' and '01100000,' with equal probability. Clearly, we need only 1 bit to distinguish one sequence from the other, so an optimal descriptive language $L_2$ would be

'00000110' → '0'
'01100000' → '1'.

Thus, on average, the optimal encoding of an input string containing $n$ symbols (or $8n$ bits) requires only $n$ bits. The efficiency of the optimal encoding is therefore $1 - 1/8 = 0.875$.

The above is an example of the more general case of a Markov random process (MRP), where $P(x_i)$ depends on $\{x_{i-1}, x_{i-2}, \ldots, x_{i-m}\}$, but is conditionally independent of the other input symbols. The optimal descriptive language for an MRP is no longer a straightforward mapping of each symbol (or group of symbols) to a unique bit string. The technical details are not important here, but the basic idea is that the coding scheme for $x_i$ becomes a function of $\{x_{i-1}, x_{i-2}, \ldots, x_{i-m}\}$. Nonetheless, the code is still unique and the number of bits required to encode $x_i$ is still $-\log_2 P(x_i)$, but $P(x_i)$ now depends on $\{x_{i-1}, x_{i-2}, \ldots, x_{i-m}\}$ instead of being fixed *a priori*. For example, one begins by encoding the first $m$ symbols in the straightforward fashion of Example 1, then one encodes the next symbol using the code defined by these $m$ symbols, then the next symbol using the code defined by the previous $m$ symbols, and so on. (Actually, the first $m$ symbols can be encoded more efficiently by taking advantage of the dependence of the second symbol on the first, and so on.) Thus, the decoder must know the encoding scheme for (or, equivalently, the probability distribution of) every possible combination of $m$ symbols. This makes the decoder more complex, of course, but allows for the optimal encoding of MRPs.

Here we saw that higher-order statistics (the conditional probability of occurence of an input symbol given a subset of the others) allowed us to capture more of the structure of the input than simple first-order statistics, at least for MRPs. In the next example we see that, in general, one must have a much deeper understanding of how the input is generated to design an optimal descriptive language and thereby fully capture the structure of the input.

### 3.5 Example 3: Independent Processes

Finally, consider a case which is much closer to the image partitioning (and more general scene analysis) problem, one where the input is a function of several independent processes. For this example, the input $\mathbf{y}$ is a function of three independent vectors, $\mathbf{l}$, $\mathbf{h}$, and $\mathbf{r}$,

$$\mathbf{y} = \mathbf{F}(\mathbf{l}, \mathbf{h}, \mathbf{r}).$$

The first two vectors $\mathbf{l}$ and $\mathbf{h}$ are combined to form a piecewise-constant vector $\mathbf{u}$ to which is added the "noise" vector $\mathbf{r}$. In other words,

$$y_i = u_i + r_i, \quad i = 0, \ldots, n-1. \tag{2}$$

The elements of $\mathbf{l}$ are the lengths of the intervals within which $\mathbf{u}$ is constant, and the elements of $\mathbf{h}$ are the heights within these intervals. For simplicity, we assume that $n$, the number of elements in $\mathbf{y}$, is fixed and is much larger than typical interval lengths. An example is illustrated in Fig. 1(a).

#### 3.5.1 Defining a Near-Optimal Descriptive Language

As stated before, the minimal-length description will require $-\log_2 P(\mathbf{y})$ bits (where $P(\mathbf{y})$ is conditioned on $n$, the known number of elements in $\mathbf{y}$). However, unlike the first two examples, there are many triples of vectors $(\mathbf{l}, \mathbf{h}, \mathbf{r})$ that map to the same input $\mathbf{y}$. Thus, the number of bits in the optimal description is

$$-\log_2 P(\mathbf{y}) = -\sum_{(\mathbf{l},\mathbf{h},\mathbf{r}):\mathbf{y}=\mathbf{F}(\mathbf{l},\mathbf{h},\mathbf{r})} \log_2 P(\mathbf{l}, \mathbf{h}, \mathbf{r}). \tag{3}$$

Because the vectors are independent of each other,

$$P(\mathbf{l}, \mathbf{h}, \mathbf{r}) = P(\mathbf{l})P(\mathbf{h})P(\mathbf{r}). \tag{4}$$

(Again, the probabilities are conditioned on $n$.) Designing an optimal descriptive language for such a process is, in general, extremely difficult and the language would almost certainly not produce stable descriptions. However, we can design a near-optimal descriptive language that produces stable descriptions for almost all inputs when the changes in height of $\mathbf{u}$ are almost always large relative to the variance of the noise elements.

When the changes in height of the constant intervals of $\mathbf{u}$ are large relative to the variance of the noise elements, there is a unique most-likely triple of vectors $(\mathbf{l}^*, \mathbf{h}^*, \mathbf{r}^*)$ that maps to $\mathbf{y}$. Furthermore, it can be shown that the probability of this vector is much larger than the sum of the probabilities of all the other vectors that map to that input. In this case, Eq. (3) reduces to

$$\begin{aligned} -\log_2 P(\mathbf{y}) &\approx -\log_2 P(\mathbf{l}^*, \mathbf{h}^*, \mathbf{r}^*) \\ &\approx -\log_2 P(\mathbf{l}^*)P(\mathbf{h}^*)P(\mathbf{r}^*) \qquad \text{(from Eq. (4))} \\ &\approx -[\log_2 P(\mathbf{l}^*) + \log_2 P(\mathbf{h}^*) + \log_2 P(\mathbf{r}^*)]. \end{aligned}$$

Thus, a near-optimal descriptive language $L_3$ is one which independently describes each of $\mathbf{l}^*$, $\mathbf{h}^*$, and $\mathbf{r}^*$ in the optimal fashion illustrated in Example 1 above.

As we shall see in the following subsections, this language is most strongly sub-optimal when the descriptions are most strongly unstable. (Roughly speaking, this occurs when the change in height from one interval to the next is small relative to the variance of the noise.) Thus, the measure of stability we shall propose tells us not only when the description is unstable but also when the language is least appropriate.

### 3.5.2 Defining the Encoder

The encoder for this example is much more complex than for the previous two examples because it must solve the optimization problem of finding the triple of vectors $(\mathbf{l}^*, \mathbf{h}^*, \mathbf{r}^*)$ that both maps to the input and requires the fewest number of bits to encode in $L_3$. A brute-force method for solving this optimization problem is to search explicitly through all possible triples, determining for each one whether it describes the input, and, if it does, comparing its encoding length to that of all other triples that describe the input.

A more elegant solution is to take direct advantage of the completeness of the description. First, note that, *for a given input*, the piecewise-constant vector $\mathbf{u}$ completely determines the residuals because, from Eq. (2),

$$r_i = y_i - u_i, \quad i = 0, \ldots, n-1.$$

Furthermore, independent of the input, the piecewise-constant function $\mathbf{u}$ implicitly specifies both $\mathbf{n}$ and $\mathbf{y}$: The lengths of the intervals wherein $\mathbf{u}$ is exactly constant are the elements of $\mathbf{l}$, and the values within these intervals are the elements of $\mathbf{h}$.[5] Thus, instead of explicitly searching through all possible triples of vectors, one need simply search through all possible vectors $\mathbf{u}$, and compute the sum of the encoding lengths of the implicitly specified vectors $\mathbf{h}$, $\mathbf{h}$, and $\mathbf{r}$.

Of course, the search space is still very large (on the order of $h^n$, where $h$ is the range of elements in $\mathbf{h}$!) if searched in a brute-force manner. However, when the elements of $\mathbf{l}$ and $\mathbf{h}$ are drawn from uniform distributions and the elements of $\mathbf{r}$ are drawn from a normal distribution, the above decomposition leads to an efficient algorithm that finds something very close to the optimal vector, $\mathbf{u}^*$, as follows.

When the elements of $\mathbf{l}$ and $\mathbf{h}$ are drawn independently from uniform distributions, the encoding length for each element is a constant, say $b_l$ and $b_h$, respectively. Thus, the combined encoding length of both vectors is simply $b = b_l + b_h$ times the number of their elements, which, as indicated above, is the number of constant intervals in $\mathbf{u}$. Furthermore, the number of constant intervals in $\mathbf{u}$ is exactly equal to one plus the number of adjacent pairs of elements $(u_i, u_{i+1})$ that are unequal. Thus, the total encoding length of $\mathbf{y}$ for a given $\mathbf{u}$ can be written as

$$|\mathcal{D}_3^{\mathbf{u}}(\mathbf{y})| = \sum_{i=0}^{n-1} |\mathcal{D}_{\mathcal{R}}(y_i - u_i)| + b + \sum_{i=0}^{n-2} b(1 - \delta(u_i - u_{i+1})), \quad (5)$$

where

$$\begin{aligned} \delta(x) &= \text{the Kronecker delta} = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise,} \end{cases} \\ |\mathcal{D}_{\mathcal{R}}(r)| &= \text{the cost of encoding a single noise element, } r. \end{aligned}$$

When noise elements are drawn from a quantized normal distribution (i.e., the elements are first drawn from $N(0, \sigma^2)$ and then rounded to the nearest $q$), the probability of drawing an element $r$ is then

$$\begin{aligned} P(r) &= \int_{\lfloor r \rfloor_q}^{\lceil r \rceil_q} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[\frac{-x^2}{2\sigma^2}\right] dx \\ &\approx \frac{q}{\sqrt{2\pi\sigma^2}} \exp\left[\frac{-r^2}{2\sigma^2}\right] \quad \text{when } q < 2\sigma. \end{aligned}$$

Thus, the cost of encoding a single element $r$ is

$$\begin{aligned} |\mathcal{D}_{\mathcal{R}}(r)| &= -\log_2 P(r) \\ &\approx \frac{1}{\log 2}\left(\frac{1}{2}\log 2\pi + \log\sigma - \log q + \frac{r^2}{2\sigma^2}\right). \quad (6) \end{aligned}$$

Substituting this into Eq. (5), we arrive at the following estimate of the encoding length

$$|\mathcal{D}_3^{\mathbf{u}}(\mathbf{f})| \approx b + nc + a\sum_{i=0}^{n-1}\left(\frac{y_i - u_i}{\sigma}\right)^2 + b\sum_{i=0}^{n-2}(1 - \delta(u_i - u_{i+1})), \quad (7)$$

where

$$\begin{aligned} a &= \frac{1}{2\log 2} \\ b &= \text{the cost of encoding an element from both } \mathbf{l} \text{ and } \mathbf{h} \\ c &= \frac{1}{\log 2}\left(\frac{1}{2}\log 2\pi + \log\sigma - \log q\right). \end{aligned}$$

Because of the Kronecker delta term, finding the optimal vector $\mathbf{u}^*$, is a complex nonlinear optimization problem. Although one can devise a straightforward dynamic programming solution for this problem (in fact, this is how we arrived at the provably optimal description of Fig. 1(c)), it is extremely difficult to generalize to the two-dimensional image partitioning problem that we present in the next section. Furthermore, the simulated-annealing style of algorithms exemplified in Geman and Geman

[5] In fact, $\mathbf{u}$ implicitly specifies more than one pair $(\mathbf{l}, \mathbf{h})$. For example, $\mathbf{u} = \{5, 5, 5, 7, 7, 7, 7\}$ not only specifies the pair $(\{3, 4\}, \{5, 7\})$ as described above, but it also specifies the pair $(\{2, 1, 4\}, \{5, 5, 7\})$ (amongst others). However, these other pairs are always more expensive to encode than the first, and so are ignored here.

[5] are also inappropriate, because the time complexity is much too high for this type of function (Blake [2]).[6]

Instead, we have devised an algorithm that yields something very close to the optimal solution for a large class of inputs and that generalizes easily to two or more dimensions. This algorithm is similar, in some respects, to that described in Blake and Zisserman [3].

To simplify the notation in the following description of the minimization algorithm, we remove constant terms and make the dependence on $\mathbf{u}$ more explicit by defining the length function

$$\begin{aligned}\mathcal{L}_3(\mathbf{u}) &= |\mathcal{D}_3^{\mathbf{u}}(\mathbf{y})| - b - nc \\ &= a\sum_{i=0}^{n-1}\left(\frac{y_i - u_i}{\sigma}\right)^2 + b\sum_{i=0}^{n-2}(1 - \delta(u_i - u_{i+1})). \quad (8)\end{aligned}$$

### 3.5.3 An Efficient Minimal-Encoding Algorithm

The difficulty with finding the global minimum of $\mathcal{L}_3(\mathbf{u})$ is that it has many local minima. Thus, standard gradient-descent based optimization techniques are useless. However, a particular class of optimization techniques, generally called continuation methods, yields something very close to or equal to the global minimum for the kind of function we have.

An intuitive description of the continuation method we shall define in the following paragraphs is that one creates a kind of "scale-space" representation of the objective function $\mathcal{L}_3(\mathbf{u})$ and tracks a local minimum from the coarsest scale (where there is only one local minimum) to the finest scale (where there are many).

More formally, the continuation method requires a family of approximations $\mathcal{L}_3^\epsilon(\mathbf{u})$ for which there is a single minimum at some large $\epsilon$ and for which the approximation converges to $\mathcal{L}_3(\mathbf{u})$ in the limit as $\epsilon$ approaches zero. We start at the unique local minimum for large $\epsilon$ and track that local minimum as $\epsilon$ approaches zero.

The specific approximation $\mathcal{L}_3^\epsilon(\mathbf{u})$ we use replaces $1 - \delta(x)$ in $\mathcal{L}_3(\mathbf{u})$ with the approximation $1 - \delta^\epsilon(x)$, where

$$\begin{aligned}1 - \delta^\epsilon(x) &= s^\epsilon(x)\left(\frac{x}{\epsilon}\right)^2 + (1 - s^\epsilon(x)) \\ s^\epsilon(x) &= \begin{cases} 1 & \text{if } |x| < \epsilon \\ 0 & \text{otherwise.}\end{cases}\end{aligned}$$

(Simply put, $1 - \delta^\epsilon(x)$ is a quadratic in $x$ when $|x| < \epsilon$, and 1 when $|x| \geq \epsilon$, see Fig. 2 for a graphical representation of this function.) In other words,

$$\begin{aligned}\mathcal{L}_3^\epsilon(\mathbf{u}) &= a\sum_i\left(\frac{y_i - u_i}{\sigma}\right)^2 \\ &\quad + b\sum_i s^\epsilon(u_i - u_{i+1})\left(\frac{u_i - u_{i+1}}{\epsilon}\right)^2 \\ &\quad + (1 - s^\epsilon(u_i - u_{i+1})). \quad (9)\end{aligned}$$

Intuitively, we can view each of the quadratic terms in Eq. (9) as representing the energy of an ideal spring, because the energy of an ideal spring is a quadratic function of its length. Moreover, because of the nonlinear term $s^\epsilon(x)$, we can view $1 - \delta^\epsilon(x)$ as representing the energy of a nonlinear spring; one which behaves like an ideal spring with spring constant $1/\epsilon^2$ until it is stretched beyond its maximum length $\epsilon$, at which point it breaks (thereafter having a constant energy of 1). In other words, we can view Eq. (9) as representing the energy in a system comprising both ideal and nonlinear springs as illustrated in Fig. 3.

In terms of this intuitive interpretation, the continuation method starts with weak nonlinear springs (small spring constants $1/\epsilon^2$), then continuously strengthens them (and continuously breaks/attaches those that become longer/shorter than $\epsilon$) until $\epsilon$ is sufficiently close to zero. As $\epsilon$ approaches zero, pairs of samples $u_i$ and $u_{i+1}$ that are held together by unbroken springs become arbitrarily close to each other, and those that have broken springs are independent of each other. Thus, in the limit as $\epsilon$ approaches zero, $\mathbf{u}$ becomes a piecewise-constant vector.

To make the continuation method discrete, we must define the order in which the nonlinear springs are adjusted for each discrete time index $t$: first decrease $\epsilon$, then strengthen the springs, then adjust their lengths to attain a global minimum in the energy (as if the springs were linear for the moment), and finally break/attach the springs according to their new lengths. Fig. 4 shows the result of this discretized continuation method, defined formally in the following paragraphs, for several values of $t$. Note that, for this example, the final result is the same as the optimal description of Fig. 1(c).

More formally, we define a sequence of quadratic forms $\mathcal{L}_3^t(\mathbf{u})$ in which the nonlinear springs of $\mathcal{L}_3^\epsilon(\mathbf{u})$ have been replaced by ideal springs whose break/attach states at time $t$ are defined by the global energy minimum of the springs at time $t - 1$. This recursive definition is grounded at $t = 0$ by choosing a sufficiently large $\epsilon$ for which we can prove that the nonlinear springs are all attached at the global energy minimum. Since the global minimum must clearly satisfy

$$y_{\min} \leq u_i^* \leq y_{\max} \quad \text{for all } i \text{ and } \epsilon,$$

where $y_{\min}$ is the smallest element in $\mathbf{y}$ and $y_{\max}$ is the largest, choosing $\epsilon > (y_{\max} - y_{\min})$ ensures that $s^\epsilon(u_i^* - u_{i+1}^*) = 1$ for all $i$. That is, all the nonlinear springs are attached at the global energy minimum of $\mathcal{L}_3^\epsilon(\mathbf{u})$ when $\epsilon > (y_{\max} - y_{\min})$.

Therefore, let

$$\begin{aligned}s^0 &= \frac{(y_{\max} - y_{\min})}{\sigma} \\ \epsilon^0 &= s^0\sigma \\ \mathcal{L}_3^0(\mathbf{u}) &= a\sum_i\left(\frac{y_i - u_i}{\sigma}\right)^2 + b\sum_i\left(\frac{u_i - u_{i+1}}{\epsilon^0}\right)^2.\end{aligned}$$

Then, for $t \geq 1$, let

$$\mathcal{L}_3^t(\mathbf{u}) = a\sum_i\left(\frac{y_i - u_i}{\sigma}\right)^2 + b\sum_i\left(d_i^t\left(\frac{u_i - u_{i+1}}{\epsilon^t}\right)^2 + (1 - d_i^t)\right),$$

where

$$\begin{aligned}\mathbf{u}^{t-1} &= \text{the unique minimum of } \mathcal{L}_3^{t-1}(\mathbf{u}) \\ d_i^t &= \begin{cases} 1 & \text{if } |u_i^{t-1} - u_{i+1}^{t-1}| < \epsilon^{t-1} \\ 0 & \text{otherwise}\end{cases} \\ s^t &= r s^{t-1} \quad (0 < r < 1) \\ \epsilon^t &= s^t\sigma.\end{aligned}$$

[6] Blake has shown, using Monte-Carlo simulations, that the time required to find the global minimum of functions similar to that of Eq. (7) increases exponentially as the "rigidity" of neighborhood constraints increases. The neighborhood constraint represented by the term $\delta(u_i - u_{i+1})$ is, in effect, an infinitely rigid constraint.

Please note that $\mathcal{L}_3^t(\mathbf{u})$ is indeed a quadratic form because the function $s^t(u_i - u_{i+1})$ of $\mathcal{L}_3^t(\mathbf{u})$ has been replaced by the set of constants $d_i^t$. In other words, $\mathcal{L}_3^t(\mathbf{u})$ represents the energy of a system comprising only ideal springs plus a constant energy term.

### 3.5.4 Computational Expense

The bulk of the computational expense in the above procedure lies in minimizing the quadratic form $\mathcal{L}_3^t(\mathbf{u})$. This can be done by solving the set of simultaneous linear equations

$$\begin{aligned}\frac{\partial \mathcal{L}_3^t(\mathbf{u})}{\partial u_m} &= 0 \\ &= \frac{2a}{\sigma^2}(u_m - y_m) + 2d_m^t \frac{b}{(s^t\sigma^t)^2}(u_m - u_{m-1}) \\ &\quad + 2d_{m+1}^t \frac{b}{(s^t\sigma^t)^2}(u_m - u_{m+1}).\end{aligned}$$

Dividing by $\frac{2b\sigma^2}{(s^t)^2}$ yields

$$-d_m^t u_{m-1} + (d_m^t + d_{m+1}^t + \frac{a}{b}(s^t)^2)u_i - d_{m+1}^t u_{m+1} = \frac{a}{b}(s^t)^2 y_m$$

(with $d_{-1}^t \equiv d_n^t \equiv u_{-1} \equiv u_n \equiv 0$).

We can solve this set of equations relatively cheaply because the matrix is banded, symmetric, and diagonally dominant, allowing for such local, parallel, and iterative solution techniques as the Gauss-Jordan iterative solution procedure. Furthermore, the solution typically differs only slightly from $t$ to $t+1$, so that the iterative solution procedure can use the previous solution as a starting point, thereby typically taking only one or a few iterations.

We have so far studiously ignored the question of a stopping criterion. That is, what value of $s^t$ is sufficiently close to zero that we can expect no further changes in the break/attach states $d_i^t$? The answer is related to the stability of the description, which is explored in the following section.

### 3.5.5 Stability of the Optimal Description

The above defined a near-optimal descriptive language and an optimal encoder for this language, but what of the stability criterion? Recall that stability is defined as the invariance of one portion of a description with respect to changes in the input that are reflections of changes to other portions of the description. Furthermore, the choice of which portions should be stable with respect to what other portions is task-dependent.

Suppose that for the piecewise-constant functions of above we are interested in the stability of the lengths of the intervals (i.e., the positions of the breaks) with respect to changes in the heights and/or the noise elements. To make the argument concrete, consider the piecewise-constant function in Fig. 1(a), whose optimal encoding is illustrated in Fig. 1(c). Now, if we change the height of the first interval by a small amount, as in Fig. 5(a), or alter the noise elements slightly, as in Fig. 5(b), the intervals of the optimal description remain the same. Thus, the optimal description for this input string is stable with respect to these types of changes.

However, when the height of the first interval is too close to that of the second, as in Fig. 5(c), arbitrarily small changes in the height of either the first or second interval, or arbitrarily small changes in the noise elements, can change the intervals of the optimal description. Clearly the optimal description for this input string is not a stable description, at least for the first two intervals. (Recall from Section 3.5.1 that this is precisely the case when the language is most strongly sub-optimal.)

This definition of stability can be directly related to the stopping criterion of our optimization algorithm. In the first example above, we find that all of the breaks are found by the algorithm well before $s^t$ approaches 1, whereas in the second example, $s^t$ must be reduced well below 1 before the first break is found, if it is found at all. (This argument has been verified for many different input strings.) Consequently, we can be fairly certain that breaks found by the algorithm will be stable with respect to small changes in the height or noise elements if they are found when $s^t$ is significantly greater than 1. In other words, a measure of the stability of a break is the value $S_i$ of $s^t$ at which it was found: when $S_i$ is much larger than 1, the break is very stable, whereas if it is much smaller than 1, it is very unstable.

Thus, a reasonable stopping criterion that yields stable descriptions is to stop when $s^t$ is close to one. Once we have found the breaks in this manner, we can determine the optimal height within each interval in a straightforward manner, because it is simply the average of the $y_i$ within the interval.

When the description will be used as a basis for further processing of the input string, it may be more useful to stop at $s^t$ much smaller than one and let other processes use the break-by-break stability measure, $S_i$, as they see fit.

The stability measure just defined is also a good measure of the stability of the positions of the breaks with respect to changes in the parameter $b$ of Eq. 7. In other words, large values of $S_i$ indicate that the breaks are stable with respect to changes in this aspect of the descriptive language. This is especially important when we do not have a precise model for the generating processes, as in the case of real images.

Of course, these arguments relating the stopping criterion to a measure of the stability of the intervals are only applicable when the descriptive language is appropriate for the input. In the next subsection, we see how we can get a reasonable measure of the appropriateness of the language once we have found the breaks and the interval heights as above.

### 3.5.6 Using Stability to Determine Appropriateness

As mentioned earlier, the stability measure should allow us to determine when a given descriptive language is inappropriate for a given input.

To illustrate, consider the noisy ramp of Fig. 6(a). Applying the optimization algorithm (where $\sigma^2$ is the true value of the variance used to generate this function) and stopping at $s^t = 1$ produces the results illustrated in Fig. 6(b). One can immediately determine that the language is inappropriate by comparing the measured variance in each interval (in this example there is only one) to the value of the variance used in the optimization. Since the measured variance is significantly different from that value, this is a strong indication that the language is inappropriate.

Alternately, we can stop the process at a much smaller value of $s^t$ and note that all the breaks have a low stability measure, as illustrated in Fig. 6(c).

Again, measures of appropriateness such as these are only computationally inexpensive approximations to the more gen-

eral method of comparing the encoding length across different descriptive languages. For example, if we were to describe this function using a piecewise-linear model (a generalization we describe in the next section), we would see that the encoding length is significantly smaller than for any piecewise-constant description.

### 3.6 Summary

In summary, we have shown that a reasonable definition of a best complete and stable description of an input string is the minimal-length information-preserving description, followed by simple tests for stability and efficiency. We have shown how to design some classes of optimal descriptive languages and have described a computationally effective procedure for finding the optimal description. We have shown that the minimal-length information-preserving description of the input is a stable description for a large class of inputs. Finally, we have described several ways of measuring the stability of the minimal-length description and indicated how this can be used to determine whether or not the descriptive language is appropriate for a given input.

## 4 A Descriptive Language for Image Partitioning

Image partitioning is a very broad problem that is unlikely to be solved in its entirety with the choice of a single simple descriptive language [4,11]. However, the language I am about to describe provides an intuitively satisfying solution to the partitioning problem for a wide variety of images.

The language is based on a world model in which all objects are composed of piecewise-differentiable surfaces whose albedo varies in a piecewise-differentiable manner. Generally speaking, discontinuities in the depth, orientation, albedo, and illumination of the surfaces will produce discontinuities in the intensities of *ideal* images taken with perfect pin-hole cameras. These image discontinuities are a good candidate for a stable component of the description of the ideal image because they remain largely invariant to small changes in depth, orientation, albedo, and illumination. Therefore, by dividing the description of an ideal image into two parts, a description of region boundaries and a description of the continuous and differentiable intensity variation within each region, one is likely to have a description in which the first part and some aspects of the second part (such as certain relations between the derivatives of intensity across discontinuities [7,8]) are stable, satisfying the stability criterion of Section 2.

Unfortunately, we can never directly observe ideal images. Instead, we must deal with real images, which we model as ideal images that have been corrupted by the imaging process. In particular, we model real images as ideal images that have been blurred, corrupted by white noise, sampled at some finite sampling rate, and quantized. This is a reasonably good model for images of the objects described above that are all at about the same depth of field. In other words, we model a real image as a finite set of observations $\{y_i = f(x_i^0, x_i^1), i = 0, \ldots, n-1\}$, where

$$f(x^0, x^1) = \left\lfloor I(x^0, x^1) * G(x^0, x^1) \right\rfloor_q + \left\lfloor R(x^0, x^1) \right\rfloor_q, \qquad (10)$$

and $(x^0, x^1)$ are the spatial coordinates of the image, $x_i^0 = i\Delta x^0$, $x_i^1 = i\Delta x^1$, $I(x^0, x^1)$ is the ideal image, $G(x^0, x^1)$ is the known point-spread function, and $R(x^0, x^1)$ is additive independent and identically distributed noise with a known distribution. Without loss of generality, we assume that $\Delta x^0 = \Delta x^1 = 1$ in the remainder of the paper.

In the following subsections, we define a series of increasingly complex descriptive languages that are increasingly better approximations to this model of corrupted images. This is done for two reasons. The first is that it is easier to understand the most complex language by first understanding the simpler ones. The second is that this sequence of languages, each being a generalization of the previous, illustrates the process of augmenting a descriptive language in order to capture more complex aspects of the images we are describing.

### 4.1 Piecewise-Constant Ideal Images

When the ideal image is piecewise-constant and the point-spread function is neglible, one can immediately see the resemblance of Eq. (10) to Eq. (2) of Example 3 when we rewrite Eq. (10) as

$$f(x_i^0, x_i^1) = u(x_i^0, x_i^1) + r(x_i^0, x_i^1),$$

where $\mathbf{u} = \{u(x_i^0, x_i^1)\}$ is the spatially discrete and intensity-quantized piecewise-constant ideal image,* and the elements of $\mathbf{r} = \{r(x_i^0, x_i^1)\}$ are the quantized residuals. We adopt the simpler notation

$$y_i = u_i + r_i,$$

where $y_i = f(x_i^0, x_i^1)$, $u_i = u(x_i^0, x_i^1)$, and $r_i = r(x_i^0, x_i^1)$, in the remainder of the paper because it simplifies the notation tremendously for the more complex languages.

As in Example 3, $\mathbf{u}$ and $\mathbf{r}$ are independent and the near-optimal descriptive language simply describes each independently. And again, the encoding length for all of the residuals is the sum of the encoding length of each residual element.

However, unlike Example 3, the optimal descriptive language for $\mathbf{u}$ is more complex. In the one-dimensional example, one simply needed to describe the length of the intervals and the height of $\mathbf{u}$ within each interval, and so the encoding length was simply proportional to the number of constant intervals, which was equal to the number of unequal adjacent pairs of $\mathbf{u}$.

In the two-dimensional case, one must describe not only the height of the regions within which $\mathbf{u}$ is constant, but also their size and shape. The descriptive language we have chosen, in analogy to the one-dimensional case, is the differential chain code of the region boundaries. In other words, each region is described by its height and its boundary, and the boundary is described by an initial point $(x, y)$, initial direction (up, down, left, or right), the number of elements along the boundary, and a sequence of the changes in the direction of the boundary. For large regions, the bulk of the encoding length will reside in the specification of the sequence of direction changes, which is proportional to the length of all the region boundaries together. Since a region boundary occurs when adjacent pairs $u_i = u(x_i^0, x_i^1)$ and $u_j = u(x_j^0, x_j^1)$ are unequal, we can write an approximation to

*By spatially discrete, we mean that $u(x_i^0, x_i^1)$ represents the ideal image intensity function $I(x, y)$ within the unit square centered at $(x_i^0, x_i^1)$. By intensity-quantized, we mean that the function values have been rounded to the nearest $q$ for some specified $q$. We assume that $q$ is small enough that all significant information has been preserved. In particular, we assume that $q$ is significantly smaller than the variance of the noise.

the total encoding length as

$$\mathcal{L}_4(\mathbf{u}) = a\sum_i \left(\frac{y_i - u_i}{\sigma}\right)^2 + b\sum_i \sum_{j\in N_i} (1 - \delta(u_i - u_j)), \quad (11)$$

where $N_i$ is the neighborhood set for index $i$ (i.e., the set of indices $j$ such the $u_j$ is adjacent to $u_i$), $a = 1/2\log 2$, $b$ is one-half the number of bits per chain-code element (since this way of writing the equation counts each element twice), and all constant expressions have been removed.

As in the one-dimensional case, one can find a good estimate of the global minimum of $\mathcal{L}_4(\mathbf{u})$ by defining a sequence of quadratic forms:

$$\begin{aligned}\mathcal{L}_4^t(\mathbf{u}) &= a\sum_i \left(\frac{y_i - u_i}{\sigma}\right)^2 \\ &\quad + b\sum_i \sum_{j\in N_i} \left(d_{i,j}^t \left(\frac{u_i - u_j}{\epsilon^t}\right)^2 + (1 - d_{i,j}^t)\right), \quad (12)\end{aligned}$$

where

$$\begin{aligned}\mathbf{u}^{t-1} &= \text{the unique minimum of } \mathcal{L}_4^{t-1}(\mathbf{u}) \\ d_{i,j}^t &= \begin{cases} 1 & \text{if } |u_i^{t-1} - u_j^{t-1}| < \epsilon^{t-1} \\ 0 & \text{otherwise} \end{cases} \\ s^t &= r s^{t-1} \quad (0 < r < 1) \\ \epsilon^t &= s^t \sigma,\end{aligned}$$

and the recursion is grounded at $t = 0$ by defining $\epsilon^0 \equiv (y_{\max} - y_{\min})$ and $d_{i,j}^0 \equiv 1$.

## 4.2 Piecewise-Polynomial Ideal Images

An approximation to the class of piecewise-smooth ideal images is the class of piecewise-polynomial images, i.e., images composed of regions whose intensity function is a two-dimensional polynomial. Of course, piecewise-constant images are a subset of such images: those for which the order of the polynomials is strictly 0.

To understand how one could construct efficient encodings for such images, consider the problem of encoding one-dimensional piecewise-polynomial functions. In a similar fashion as for the one-dimensional piecewise-constant case, we divide the function into intervals, where the intensity function within each interval can now be represented by a single polynomial of finite order. If we were to encode each interval independently, the encoding length for each interval would be proportional to the number of non-zero coefficients of the polynomial. However, we can reduce the encoding length even further by taking advantage of the relationship we can expect to find between the polynomials in adjacent intervals. For example, if we encode the intervals from left to right, and if the function is continuous (but not differentiable) at the first interval boundary, one can take advantage of the continuity to use one less coefficient to describe the polynomial in the second interval (i.e., the polynomial in the second interval has one less degree of freedom than it would otherwise have).

To summarize, efficient encodings of one-dimensional piecewise-polynomial functions demand two things. First is that the number of non-zero coefficients in each interval should be as small as possible, and second is that the number of coefficients that are different between adjacent intervals should be as small as possible.

Similarly, efficient encodings of piecewise-polynomial images demand two things. First is that the number of non-zero coefficients in each region should be as small as possible, and the second is that the number of coefficients that are different between neighboring regions should be as small as possible.

As for the piecewise-constant case, we represent the piecewise-polynomial image in a spatially discrete manner. However, we replace the scalar $u_i$ (representing the constant intensity function within the unit square centered at $(x_i^0, x_i^1)$) with a vector $\mathbf{u}_i = \{u_{i,k},\ k = 0, \ldots, m-1\}$ (representing the polynomial intensity function within the unit square). The particular vector we use is the vector of coefficients of the Taylor expansion of the polynomial about the center point $(x_i^0, x_i^1)$. In other words, the intensity function within the unit square is of the form

$$\begin{aligned}&u_{i,0} + u_{i,1}(x^0 - x_i^0) + u_{i,2}(x^1 - x_i^1) + u_{i,3}(x^0 - x_i^0)^2 \\ &+ u_{i,4}(x^0 - x_i^0)(x^1 - x_i^1) + u_{i,5}(x^1 - x_i^1)^2 + \ldots.\end{aligned}$$

As previously mentioned, the first term in the encoding length for each region is proportional to the number of non-zero polynomial coefficients in that region. Moreover, in order to maintain reasonable accuracy over an entire region, one would expect that the number of bits required to encode each non-zero coefficient would increase with region size. Rissannen [10] shows that, for a particular encoding scheme for one-dimensional polynomials, the number of bits increases with the logarithm of the number of samples. In order to maintain the locality of our approximation, we will assume that the the number of bits increases linearly with the number of non-zero coefficients. Thus, the first term in our estimate of the encoding length is

$$d\sum_i \sum_k (1 - \delta(u_{i,k})), \quad (13)$$

where $d$ is the number of bits required to encode a non-zero coefficient, divided by the average region size.

The second term in the encoding length is the number of coefficients that are different between neighboring regions. As in the piecewise-constant case, we compute this number locally by counting the number of additional coefficients that we need in order to represent the combined intensity function of the two adjacent pixels. For the piecewise-constant case, this number was $(1 - \delta(u_i - u_j))$. That is, the constant intensity functions were either the same, requiring no additional coefficients, or they were different, requiring one additional coefficient.

One way of computing this number for a pair of neighboring pixels $(x_i^0, x_i^1)$ and $(x_j^0, x_j^1)$ is to compare the polynomial and its derivatives at some common point, such as the average of the point coordinates, $(\frac{x_i^0 + x_j^0}{2}, \frac{x_i^1 + x_j^1}{2})$. Each non-zero difference in the polynomial or any of its derivatives means that an additional coefficient is required. For example, if we restrict ourselves to first-order polynomials for the moment, the number of additional coefficients required to represent the combined intensity function for a pixel $(x_i^0, x_i^1)$ and its neighbor to the right $(x_j^0, x_j^1) = (x_i^0 + 1, x_i^1)$ is

$$\begin{aligned}&\{1 - \delta([u_{i,0} + 0.5u_{i,1}] - [u_{j,0} - 0.5u_{j,1}])\} \\ &+ \{1 - \delta(u_{i,1} - u_{j,1})\} \\ &+ \{1 - \delta(u_{i,2} - u_{j,2})\}.\end{aligned}$$

In general, the polynomial or any of its derivatives at some common point can be expressed as a linear combination of the Taylor

coefficients, so that the general form for counting the number of additional coefficients is

$$\sum_k \left(1 - \delta\left(\left[\sum_l p_{i,j,k,l}\, u_{i,l}\right] - \left[\sum_l p_{j,i,k,l}\, u_{j,l}\right]\right)\right).$$

Thus, the second term in the encoding length is

$$b \sum_i \sum_{j \in N_i} \sum_k (1 - \delta(P_{i,j,k} - P_{j,i,k})), \tag{14}$$

where

$$P_{i,j,k} = \sum_l p_{i,j,k,l}\, u_{i,l}$$
$$P_{j,i,k} = \sum_l p_{j,i,k,l}\, u_{i,l}.$$

Combining Eq. (13) and Eq. (14) with the term for the encoding length of the residuals, we arrive at the following approximation for the total encoding length:

$$\begin{aligned} \mathcal{L}_5(\mathbf{u}) &= a \sum_i \left(\frac{y_i - u_{i,0}}{\sigma}\right)^2 \\ &\quad + b \sum_i \sum_{j \in N_i} \sum_k (1 - \delta(P_{i,j,k} - P_{j,i,k})) \\ &\quad + d \sum_i \sum_k (1 - \delta(u_{i,k})). \end{aligned} \tag{15}$$

Finally, we can include the point-spread function by approximating the convolution with the ideal piecewise-polynomial image by a discrete convolution with the spatially discrete piecewise-polynomial image, yielding

$$\begin{aligned} \mathcal{L}_6(\mathbf{u}) &= a \sum_i \left(\frac{y_i - \sum_{j \in B_i} \sum_k G_{i,j,k}\, u_{j,k}}{\sigma}\right)^2 \\ &\quad + b \sum_i \sum_{j \in N_i} \sum_k (1 - \delta(P_{i,j,k} - P_{j,i,k})) \\ &\quad + d \sum_i \sum_k (1 - \delta(u_{i,k})). \end{aligned} \tag{16}$$

where $G_{i,j,k}$ are the weights of the discrete convolution kernel, and $B_i$ is the the spatial support of $G$, i.e., the set of indices $j$ where $G_{i,j,k}$ is non-zero for at least one $k$.

As always, we solve for the global minimum by defining a sequence of quadratic forms.

### 4.3 Images with Known Spatially Varying Noise

The discussion so far has assumed that the variance of the noise is both constant and known *a priori*. We can deal with known variance that is different from point to point simply by changing $\sigma$ to $\sigma_i$ in the first summation of the encoding length function. Thus, for example, $\mathcal{L}_4(\mathbf{u})$ becomes the encoding length function

$$\mathcal{L}_7(\mathbf{u}) = a \sum_i \left(\frac{y_i - u_i}{\sigma_i}\right)^2 + b \sum_i \sum_{j \in N_i} (1 - \delta(u_i - u_j)). \tag{17}$$

The sequence of quadratic forms required to minimize this function is slightly more complex, in part because of the criterion that the sequence is stopped at $t$ when $\epsilon^t \approx \sigma$. Thus, when $\sigma$ is different from point to point ($\sigma_i$), $\epsilon$ must also be different from point to point because it must never fall below $\sigma_i$ at any point. Furthermore, since $\epsilon$ represents the spring-length between neighboring points $i$ and $j$, it must be defined in a symmetric fashion. The sequence of quadratic forms we have defined that satisfies these requirements is

$$\begin{aligned} \mathcal{L}_7^t(\mathbf{u}) &= a \sum_i \left(\frac{y_i - u_i}{\sigma_i}\right)^2 \\ &\quad + b \sum_i \sum_{j \in N_i} \left(d_{i,j}^t \left(\frac{u_i - u_j}{\epsilon_{i,j}^t}\right)^2 + (1 - d_{i,j}^t)\right), \end{aligned}$$

(compare this with Eq. (12)) where

$$\begin{aligned} \mathbf{u}^{t-1} &= \text{the unique minimum of } \mathcal{L}_7^{t-1}(\mathbf{u}) \\ d_{i,j}^t &= \begin{cases} 1 & \text{if } |u_i^{t-1} - u_j^{t-1}| < \epsilon_{i,j}^{t-1} \\ 0 & \text{otherwise} \end{cases} \\ s^t &= r\, s^{t-1} \quad (0 < r < 1) \\ \epsilon_i^t &= s^t \sigma_i \\ \epsilon_{i,j}^t &= \max(\epsilon_i^t, \epsilon_j^t), \end{aligned}$$

and the recursion is grounded at $t = 0$ by defining $d_{i,j}^0 \equiv 1$ and $s^0$ such that the smallest $\epsilon_i^0$ equals $(y_{\max} - y_{\min})$. As before, $s^t$ is a good measure of the stability of a break. Note that this definition of $\mathcal{L}_7^t(\mathbf{u})$ reduces to $\mathcal{L}_4^t(\mathbf{u})$ when the $\sigma_i$s are constant.

### 4.4 Images with Unknown Spatially Varying Noise

But what if we don't know the $\sigma_i$s? Then, according to the minimal encoding-length criterion, we must find those values of the $\sigma_i$s that minimize the overall encoding length, *including the cost of encoding the $\sigma_i$s themselves in some descriptive language*.

One possible model for spatially varying noise is that the variance is piecewise-constant, with the variance boundaries coinciding with the intensity boundaries. Thus, for this model, the cost for encoding a region boundary will now be slightly higher (since the cost of the boundary subsumes the cost of encoding the parameters within the region), and we need to include a term which ensures that $\sigma_i = \sigma_j$ for all adjacent pairs not crossing a region boundary. Again, using the piecewise-constant-intensity model for simplicity, and reinserting the term involving $\log \sigma$ that was removed for convenience when $\sigma$ was fixed (see the definition of $a$ and $c$ in eq. (7)), the encoding length function becomes

$$\begin{aligned} \mathcal{L}_8(\mathbf{u}, \sigma) &= \frac{1}{\log 2} \sum_i \left(\frac{1}{2}\left(\frac{y_i - u_i}{\sigma_i}\right)^2 + \log \sigma_i\right) \\ &\quad + b \sum_i \sum_{j \in N_i} (1 - \delta(u_i - u_j)) \\ &\quad + c \sum_i \sum_{j \in N_i} \delta(u_i - u_j)(1 - \delta(\sigma_i - \sigma_j)), \end{aligned} \tag{18}$$

where $b$ is now slightly larger than for $\mathcal{L}_4(\mathbf{u})$, and $c \gg b$.

One could attempt to find the global minimum of this function by defining a sequence of functions in the same vein as before. Unfortunately, these functions would not be quadratic forms because of the $\log \sigma_i$ term. Thus, one would be faced with the problem of finding the global minimum of each of these functions in turn.

Instead, we use the following line of reasoning. First, observe that at the global minimum $(\mathbf{u}^*, \sigma^*)$, and for a sufficiently large $c$, $\sigma_i^* = \sigma_j^*$ whenever $u_i^* = u_j^*$. That is, $\sigma^*$ is constant within the contiguous regions that $\mathbf{u}^*$ is constant, as we demanded. To

make this explicit, let $R_r$ denote the set of indices within the $r^{th}$ region, and let $u_r$ and $\sigma_r$ denote the constant values of $\mathbf{u}^*$ and $\sigma^*$ within this region. Thus, we can rewrite eq. (18) at the global minimum as

$$\mathcal{L}_8(\mathbf{u}^*,\sigma^*) = \frac{1}{\log 2}\sum_r \sum_{i\in R_r}\left(\frac{1}{2}\left(\frac{y_i-u_r}{\sigma_r}\right)^2+\log\sigma_r\right) + b\sum_r \text{boundary length of } R_r. \quad (19)$$

Since the boundaries are fixed by definition, eq. (19) is minimal when each term

$$\sum_{i\in R_r}\left(\frac{1}{2}\left(\frac{y_i-u_r}{\sigma_r}\right)^2+\log\sigma_r\right)$$

is itself minimal. Since this term is differentiable, we can determine the values $u_r$ and $\sigma_r$ that minimize it by differentiating and setting to zero, which yields

$$u_r = \frac{1}{n_r}\sum_{i\in R_r} y_i$$
$$\sigma_r^2 = \frac{1}{n_r}\sum_{i\in R_r}(y_i-u_r)^2,$$

where $n_r$ is the number of elements in $R_r$. In other words, as we might have expected, the minimal-length encoding occurs when the intensity estimate within each region equals the region average and the variance parameter equals the region variance.

In other words, if we knew the region variance, we could minimize $\mathcal{L}_8(\mathbf{u},\sigma)$ in the same way that we minimized $\mathcal{L}_7(\mathbf{u})$, simply by substituting $\sigma_i = \sigma_r$ for $i \in R_r$.

Of course, this is not directly possible, but we can come close by noting that the region variance is approximately equal to the average of local estimates of the variance within the region; that is[8]

$$\sigma_r^2 = \frac{1}{n_r}\sum_{i\in R_r}(y_i-u_r)^2 \approx \frac{1}{n_r}\sum_{i\in R_r}\frac{\sum_{j\in N_i\cap R_r}(y_j-u_r)^2}{\sum_{j\in N_i\cap R_r}1}.$$

Thus, by adding a term to the sequence of quadratic forms that converges to the average of local estimates of the variance, we should come close to the global minimum. We start with local estimates of the variance based directly on the data, and improve these estimates from $t$ to $t+1$ by basing the local estimates on $\mathbf{u}^{t-1}$. In other words, we define the sequence of quadratic forms as follows:

$$\mathcal{L}_8^t(\mathbf{u},\sigma) = \frac{1}{2\log 2}\sum_i\left(\frac{y_i-u_i}{\sigma_i^{t-1}}\right)^2 + b\sum_i\sum_{j\in N_i}\left(d_{i,j}^t\left(\frac{u_i-u_j}{\epsilon_{i,j}^t}\right)^2+(1-d_{i,j}^t)\right) + \frac{1}{2\log 2}\sum_i\left(\frac{\hat{\sigma}_i^t-\sigma_i}{\sigma_i^{t-1}}\right)^2 + c\sum_i\sum_{j\in N_i}\left(d_{i,j}^t\left(\frac{\sigma_i-\sigma_j}{\epsilon_{i,j}^t}\right)^2+(1-d_{i,j}^t)\right),$$

where

$$(\mathbf{u}^{t-1},\sigma^{t-1}) = \text{the unique minimum of } \mathcal{L}_8^{t-1}(\mathbf{u},\sigma)$$
$$d_{i,j}^t = \begin{cases} 1 & \text{if } |u_i^{t-1}-u_j^{t-1}| < \epsilon_{i,j}^{t-1} \\ 0 & \text{otherwise}\end{cases}$$
$$\hat{\sigma}_i^t = \text{the local estimate of the standard deviation} = \sqrt{\frac{\sum_{j\in N_i} d_{i,j}^t\left(y_j-u_i^{t-1}\right)^2}{\sum_{j\in N_i} d_{i,j}^t}}$$
$$s^t = r\,s^{t-1} \quad (0<r<1)$$
$$\epsilon_i^t = s^t\sigma_i^{t-1}$$
$$\epsilon_{i,j}^t = \max(\epsilon_i^t,\epsilon_j^t),$$

and the recursion is grounded at $t=0$ by defining $d_{i,j}^0 \equiv 1$, $u_i^{-1} \equiv y_i$ (thereby defining $\sigma_i^0$ in terms of the data only), and $s^0$ such that the smallest $\epsilon_i^0 = (y_{\max} - y_{\min})$.

## 4.5 Results

The elements of the descriptive languages that we have just defined are, of course, extreme simplifications of the processes that form an image. There is no direct notion of the three-dimensional nature of objects, their interaction with illuminants, nor even any notion of texture. Yet, even such extremely simplified descriptive languages yield intuitively pleasing partitions of real images.

For example, Fig. 7 illustrates an application of the piecewise-first-order-intensity with piecewise-constant-noise model to an aerial photograph of a house. Fig. 7(b) and (c) illustrate the underlying piecewise-first-order image (i.e., the vector $\mathbf{u}$) and its discontinuities, after having stopped the process at $s^t = 0.25$. Fig. 7(d) is an image of the stability measure ($S_i$) for these discontinuities. There are two interesting points about this example. First, note that the four bushes in the lower-left corder were delineated even though the contrast along part of their boundaries is virtually nil. This is extremely difficult to do with either standard region-based or edge-based approaches. Second, note that the bulk of discontinuities that form closed regions have high stability measures. This is a fairly strong indication that the piecewise-first-order model is appropriate. If we now demand a truly piecewise-first-order description, we can remove all discontinuities that do not form closed regions, and explicitly fit first-order polynomials to the remaining regions, as illustrated in Fig. 7(e) and (f).

A second example is an application of the same model (using precisely the same parameters) to the image of a face, illustrated in Fig. 8. In this example, about half the discontinuities have a fairly low stability measure. This is an indication that the language is probably not appropriate for this image. This is especially evident in the forehead, cheek, and chin areas where a higher-order model is clearly more appropriate. Even so, the discontinuities with high stability measures appear to be good candidates for the boundaries of regions for higher-order models. Preliminary experiments with higher-order models (which could not be carried out on the full image because of time constraints) show promising results.

[8] The inequality occurs only because of boundary conditions. Thus, the approximation is best for large regions, where the effects of the boundary conditions are minimal.

## 5 Summary

We have presented a new approach to the image partitioning problem: that of constructing a complete and stable description of an image in terms of a descriptive language, that is simplest in the sense of shortest description length. We have presented criteria on which to base formal definitions of completeness, stability, and simplicity, and have applied these criteria to the theory of minimal-length descriptions. This formalism is very general and is likely to be applicable in other stages of the scene analysis process.

For the specific image-partitioning problem, we described real images as the corruption of ideal (piecewise-polynomial) images by blur and additive white noise. We defined a language for describing both the ideal image and the corruptions, and presented an algorithm for finding the simplest description of an image in terms of this language. The algorithm produces both this simplest description and a measure of the stability of the description.

Applications of this formalism to real images indicate that, even though the descriptive language we have defined is extremely simple (with no models of three-dimensional shape, lighting, or texture, for example), the simplest description in this language yields excellent image partitions.

## Acknowledgments

I wish to thank Dr. Edwin Pednault for introducing me to the topic of minimal-length encoding and for the many discussions that ensued; Dr. Demetri Terzopoulos for the many discussions we have had concerning optimization theory; and Dr. Martin Fischler for the discussions concerning the importance of criteria other than simplicity.

## References

[1] T.O. Binford, "Inferring Surfaces from Images," *Artificial Intelligence*, **17**(1-3), pp. 205–244, 1981.

[2] A. Blake, personal communication.

[3] A. Blake and A. Zisserman, *Visual Reconstruction*, MIT Press, Cambridge, Massachusetts, 1987.

[4] M.A. Fischler and R.C. Bolles, "Perceptual Organization and Curve Partitioning," *IEEE PAMI*, **8**(1), pp. 58–68, 1984.

[5] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE PAMI*, **6**(6), pp. 721–741, 1984.

[6] M.P. Georgeff and C.S. Wallace, "A General Selection Criterion for Induction Inference," *SRI Technical Note* 372, SRI Int., Menlo Park, California, 1985.

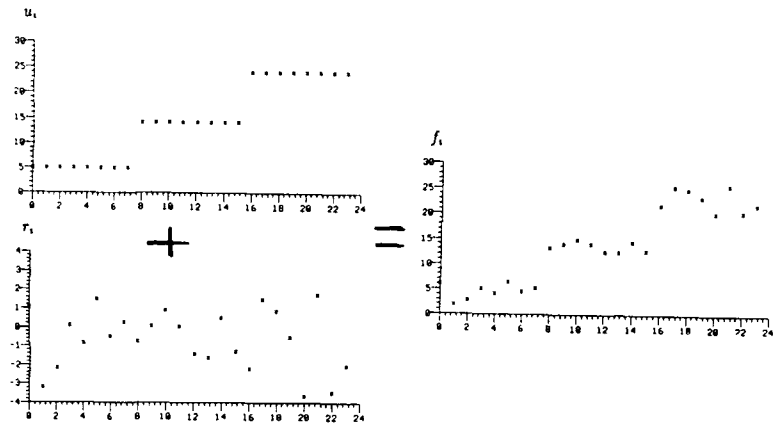
[7] Y.G. Leclerc, "Capturing the Local Structure of Image Discontinuities in Two Dimensions," in *Proc. IEEE Comput. Soc. Conf. on Comput. Vision and Pattern Recognition*, San Francisco, California, pp. 34–38, June, 1985.

[8] Y.G. Leclerc and S.W. Zucker, "The Local Structure of Image Discontinuities in One Dimension," *IEEE PAMI*, **9**(3), pp. 341–355, 1987.

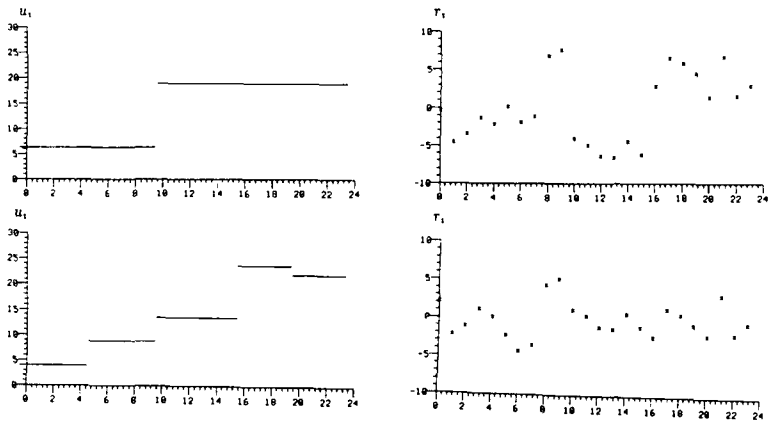
[9] J. Rissanen, "Minimum-Description-Length Principle," in *Encyclopedia of Statistical Sciences*, **5**, J. Wiley, New York, pp. 523–527, 1987.

[10] J. Rissanen, "A Universal Prior for Integers and Estimation by Minimum Description Length," *The Annals of Statistics*, **11**(2), pp. 416–431, 1983.

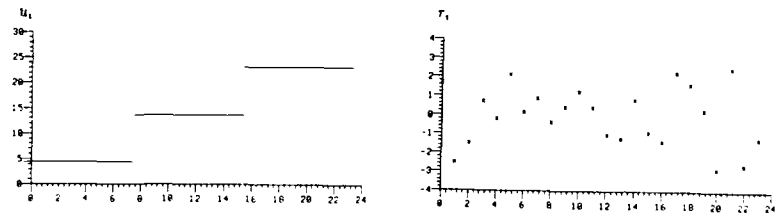
[11] S.W. Zucker, "The Diversity of Perceptual Grouping," *McGill University Computer Vision and Robotics Laboratory Technical Note* TR-85-1R, McGill University, Montréal, Québec, Canada, 1985.



(a)



(b)



(c)

Figure 1: (a) A piecewise-constant vector embedded in noise. (b) Several different descriptions that produce the same noisy vector. (c) The optimal description of the noisy vector. Note that the interval lengths are precisely the same as for the the underlying vector **u**, and that the interval heights are very close to that of **u**.

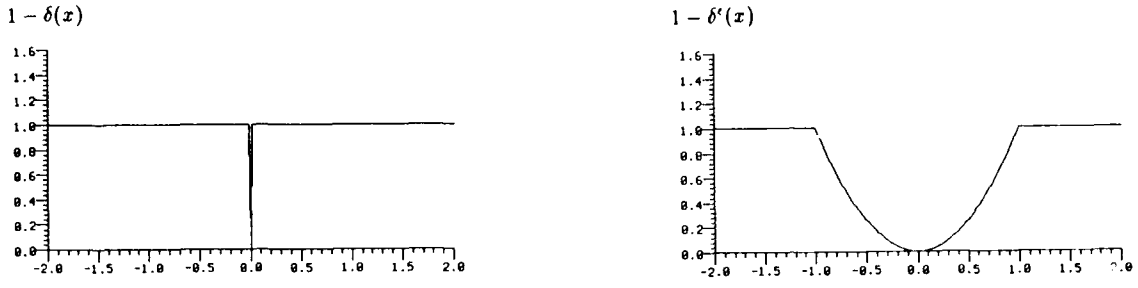



Figure 2: The graph of $1 - \delta(x)$ compared to that of $1 - \delta'(x)$.

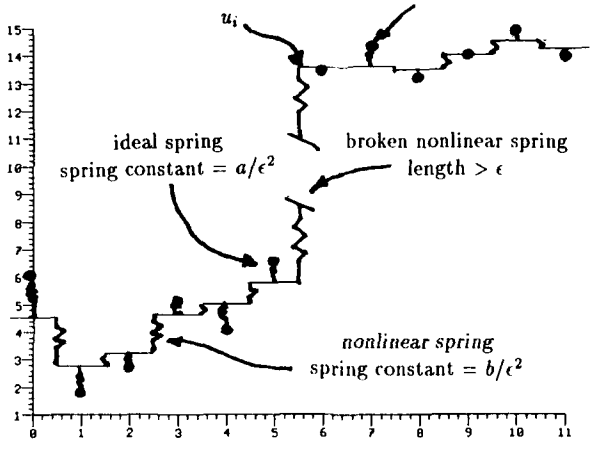



Figure 3: The system of springs whose combined energy represents the encoding-length of the input y when $\epsilon$ approaches zero.

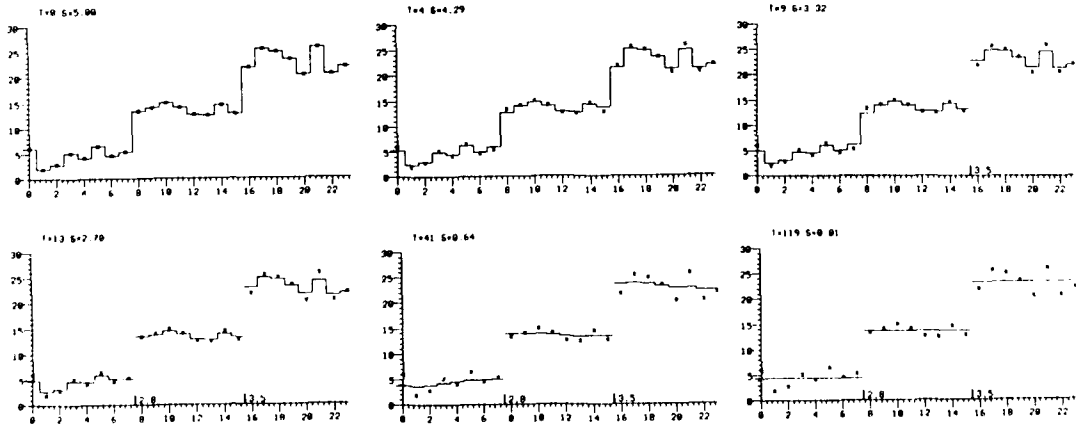


Figure 4: The state of the discrete continuation-method for several values of $t$. Note that the final state is the same as the optimal description of Fig. 1.

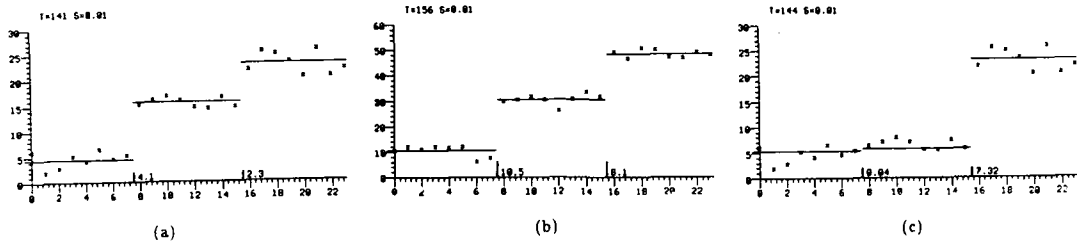


Figure 5: (a) Slightly altering the height of the first interval does not effect the optimal description. (b) Slightly altering the noise elements also does not effect the optimal encoding. (c) When the change in height between the first two intervals is too small, the optimal description is unstable.

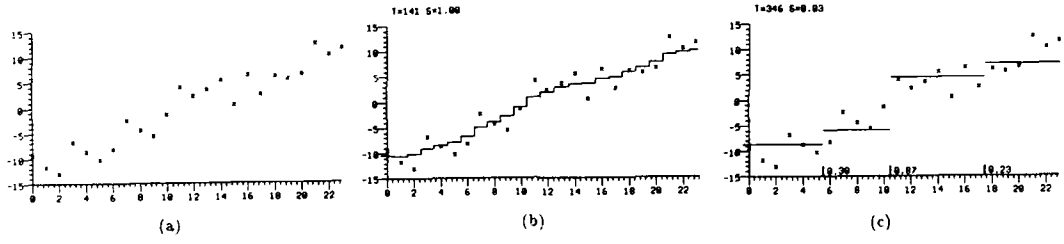


Figure 6: (a) A noisy ramp vector. (b) The result of the optimization procedure when stopped at $s^t = 1$. Note that no breaks have been found, and that the variance from the mean is clearly much larger than the variance of the noise used to generate the vector. (c) The result of the optimization procedure when stopped at $s^t = 0.1$. Note that the stability of the breaks (indicated numerically immediately above the axis) is very low.

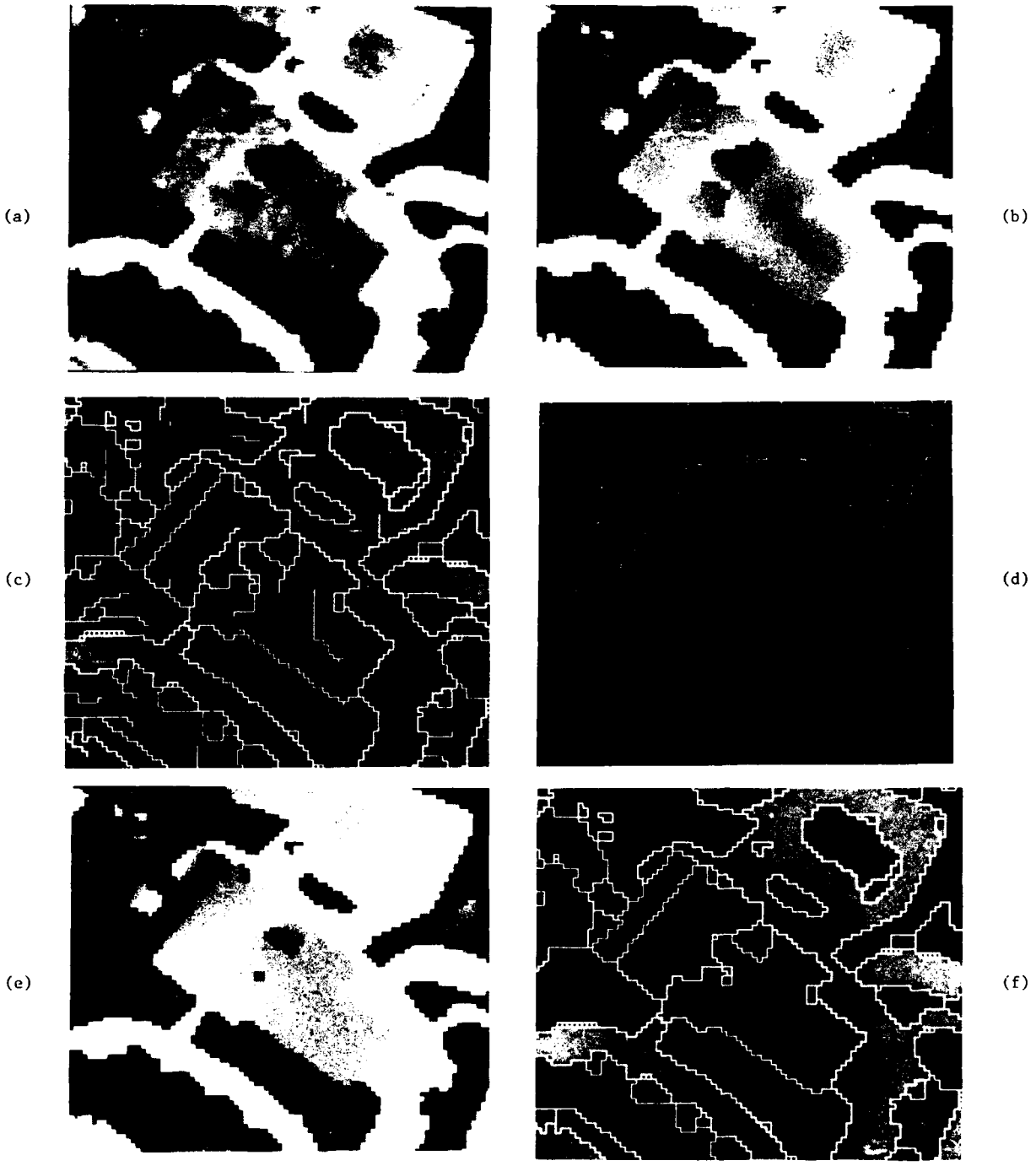
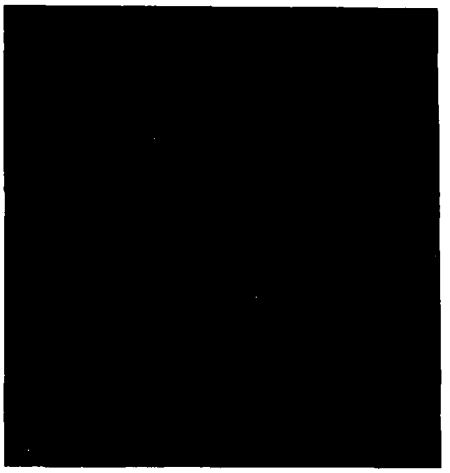



Figure 7: An application of the piecewise-first-order-intensity with piecewise-constant-noise model to an aerial photograph of a house. (a) Original image, (b) the underlying ideal image $\mathbf{u}^t$, after stopping the optimization procedure at $s^t = 0.25$, (c) the discontinuities of $\mathbf{u}^t$ overlaid on the original image, (d) the stability measure for the discontinuities (brighter indicates more stable), (e) the ideal image $\mathbf{u}^t$ after all discontinuities not forming closed regions have been removed and first-order polynomials have been fit to the remaining regions, (f) the discontinuities of this modified $\mathbf{u}^t$ overlaid on the original image.

(a)

(b)

(c)



(d)

Figure 8: An application of the same model as in Fig. 7 to the image of a face. (a) Original image, (b) the underlying ideal image $\mathbf{u}^t$, after stopping the optimization procedure at $s^t = 0.25$, (c) the discontinuities of $\mathbf{u}^t$ overlaid on the original image, (d) the stability measure for the discontinuities (brighter indicates more stable).

# 3-D OBJECT RECOGNITION USING SURFACE DESCRIPTIONS[1]

T.J. Fan, G. Medioni, and R. Nevatia

Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273

## ABSTRACT

In our previous work, we have developed a method to segment and describe range images using surface properties. In this paper, we show how these descriptions can be used to establish correspondences between 3-D objects as needed in recognition.

In a first step, we group labeled curves corresponding to jump boundaries, creases, and limbs of objects into boundaries of regions. Each region is therefore described by its boundaries, and by a polynomial approximation, which allows us to reconstruct each patch individually, and also the complete scene. The scene is represented by a *graph* with the patch as the *nodes* and relations between them as the *links*. In a second step, we infer objects (or partial objects) from surface patches, then match two range images at this partial object level. Graphs of objects are matched using a *best-first* search under three types of constraints: unary constraints between corresponding nodes, binary constraints between corresponding linked pairs of nodes, and contraints imposed by the computed geometric transformation. The matching is *data-driven* in the sense that no *a priori* information such as pre-built model is required. Substantial partial occlusion is also allowed.

The results are presented in two instances:

- *Matching two complex scenes*: Objects that appear in two images are identified without a priori information of these objects. Descriptions for both images are computed independently and a graph for each image is obtained. Then a *best first graph search* is applied to find the best match between these two images. The transformations between corresponding objects are also calculated. The objects in both images may vary in orientation and position, they can also be occluded by other objects. Various real range images have been used in the testing with very good results.

- *Multi-view model building and object recognition*: The descriptions of objects are used to build multi-view models. Each model consists of several views (6 to 8). These views are arranged so that most of the siginificant surfaces of the model object will be contained in at least one of these views. Finally, given a new scene which contains various objects, a model matching is applied, and a best model, if any, is selected for each scene object. It should be noted that more views could have been used for each model; this would however increase the search space in the matching process, and the results show that 6 to 8 views should be sufficent in general.

[1]This research was supported by the Defense Advanced Research Projects Agency under contract number F33615-87-C-1436, monitored by the Air Force Wright Aeronautical Laboratories, Darpa Order No. 3119.

## 1 Introduction

We are interested in establishing correspondences betweeen range images representing 3-D objects. Good matching techniques are needed for many generic tasks such as object inspection, recognition, and mechanical manipulation of the objects. One can match two scenes at many different levels of descriptions with some trade-offs: The lower the level of the descriptions, the easier it is to compute them. For example, a range array may be available directly as input and an Extended Gaussian Image [13] requires only the computation of surface normals. However, low level descriptions are not invariant to viewing directions and little tolerant to occlusion. The higher level descriptions, on the other hand, maintain their invariance but the algorithms to compute them are often weak and error-prone. The appropriate level of description to be used for matching thus depends on the expected variations in the scenes and on the state-of-art in computing descriptions of the scene.

We have decided, in this work, to use object descriptions in terms of their surfaces. The surface of an object is described by segmenting it into *surface patches* and the complete description consists of the description of each patch separately, and their inter-relationships. Such a description can be viewed as a *graph* with the patches as the *nodes* and relations between them as the *links*. The segmentation and description of the surface is based on measured curvature properties of the surface. We describe this process briefly in section 2, more details are given in previously reported work [7,8].

We believe that our chosen representation has many advantages for scene and object matching. The description is *rich*, so that similar objects can be identified, *stable*, so that local changes do not radically alter it, and has *local support* so that partially visible objects can be identified. It also enables us to recreate, from its features, a shape reasonably close to the original one. The surface descriptions are much higher-level than pointwise or edge descriptions, but not as high as volumetric descriptions. The surface descriptions are invariant for smaller changes in viewing angle than would be the case for volume descriptions. However,

techniques for volume descriptions are not yet fully developed, whereas our previous work allows us to have high-quality surface descriptions.

We assume that range data (i.e., 3-D positions) of the points on the visible surface are available, for example, by the use of a laser range finder. It is also assumed that this data is *dense*, in the sense of being sampled on a certain grid and not just at discontinuities (as may be the case for uninterpreted edge-based stereo data). In the remainder of this paper, the terms *scene*, *range data*, and *range image* are interchangeably used.

There has been much work in the area of object description and recognition, it is impossible for us to give a detailed survey here, good sources for survey are [1,6]. Some typical works are summarized below.

- 3DPO [4] recognizes objects in a jumble, verifies them, and then determines essential configurational information, such as which ones are on top. The system only use cylindrical and straight edges to recognize objects, hence is rather restricted in the shape of the objects.
- ACRONYM [5] is a model-based image understanding system. The user is responsible to provide complete descriptions of the models. Furthermore, the results shown are very limited in change of viewing directions.
- Faugeras and Hebert [9] developed a system to recognize and locate rigid objects in 3-D space. In their system, representations should be in terms of linear primitives, such as points, lines, and planes.
- Grimson and Lozano-Pérez [10,11] discusses how local measurements of 3-D positions and surface normals can be used to identify and locate objects from among a set of unknown objects. Only polyhedral objects are used in the testing.
- Oshima [16,17] describes an approach to the recognition of stacked objects with planar and curved surfaces. Only limited occlusion is allowed on planar surfaces, and no occlusion is allowed for curved surfaces.

Most systems are highly model-dependent and restricted to the class of models known to them. By using models, it is possible to work with low-level scene descriptions, but at a cost in flexibility and generality. We believe that our system is more general than those described as above.

The next section briefly explains how we go from a dense range map to a set of surface patches, then how these patches are further organized into partial objects. Section 3 presents the matching algorithm which uses the descriptions given above, and shows how the results of matching may be used to refine the segmentation of the scene into objects. Section 4 presents the multi-view model-based recognition. Finally, section 5 summarizes our contribution.

# 2 Computing Symbolic Scene Descriptions

In the following, we describe two major steps of our description process: the first is the extraction and description of surface patches, the second is the inference of objects from these patches.

## 2.1 Extracting and describing surface patches

Given a complex surface, we wish to decompose it into simple surface patches. Our approach is to determine the boundaries of the surfaces by computing *local* properties and then inferring the patches from them. Similar ideas for segmentation have been used in [2,3]. In particular, we seek to find the following kinds of boundaries:

1. **jump boundaries** where the surface undergoes a discontinuity
2. **creases** which correspond to surface orientation discontinuities

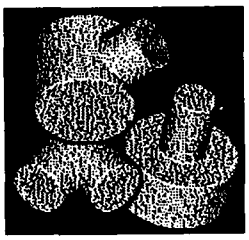
We infer these boundaries from curvature properties of the surface. In particular, we use curvature *zero-crossings* and *extrema* [7]. A *jump boundary* creates a zero-crossing of the curvature in a direction normal to that of the boundary; a *crease* causes a local extremum of the curvature at that point. Crease boundaries may also create zero-crossings away from the location of the boundary itself.

These features, when connected using contiguity, give us *partial* boundaries for patches in which the surface should be segmented but not necessarily a *complete* segmentation. These partial boundaries are then completed by simple extension [8]; we have found this process to be satisfactory for a large number of examples we have tested it on. The resulting *regions* are assumed to correspond to elementary surface patches. These regions could be segmented further, either based on the region shape or on the results of surface fitting; we have not found it necessary to do so for the examples we have tried.

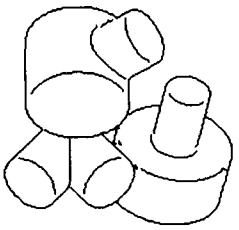
The surface patches are approximated by a second order polynomial in $(x, y, z)$, whose coefficients are computed by a least-squares method. The details are given in [8]. Figure 1 illustrates this early processing on a synthetic image for reasons of clarity, but the method has been applied with success on a large number of real scenes. Most researchers display range images by encoding depth by grey level, but this produces images with very poor dynamic range; in the rest of the paper, we present range images by borrowing a technique from computer graphics: we assume that the object is Lambertian, compute the normal to the surface in a small (3x3) neighborhood, and generate a *reflectance* image in which the intensity is inversely proportional to the angle between the light source and the surface normal. Figure 1(a) shows the shaded image representing three objects partially occluding each other, figure 1(b) shows the curves obtained by our feature detection process, figure 1(c) shows the regions bounded by the previous curves, and figure 1(d) shows the *reconstructed* shaded scene, after each patch has been approximated by a quadratic polynomial.

## 2.2 Object level descriptions

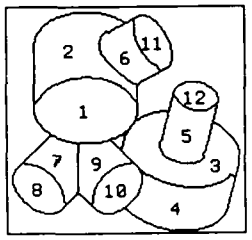
At the end of above process, we obtain a symbolic representation of a scene as a *graph* whose nodes represent the patches and whose links express geometric relationships between patches, making it an appropriate level to use in a matching process. We found, however, that it is possible to further group these patches into *objects*, or rather visible faces of objects, by reasoning on the type



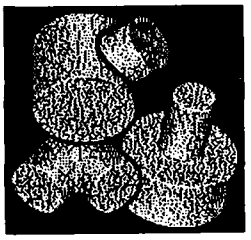
(a) Original scene



(b) Features chosen for segmentation




(c) Result of segmentaion



(d) Result of reconstruction

Figure 1: Segmentation and reconstruction.

of connections between adjacent patches. This even higher level of description facilitates the matching process. We now describe the procedure used to obtain these *objects*.

### 2.2.1 Labeling boundaries

So far, we have classified the boundaries into two classes: jump boundaries and creases. Once each patch is approximated by an analytic function, it becomes possible to distinguish true jump boundaries from *limbs* (also called axial contour generators [19]), for which the normal to the surface becomes perpendicular to the viewing direction. This distinction is very important for volume inference, and also for matching, as limbs are *not* intrinsic properties of an object, but depend on the viewing direction.

We therefore end up with the following set of four labels:

1. **convex crease (+):** It corresponds to a negative curvature extremum.
2. **concave crease (−):** It corresponds to a positive curvature extremum.
3. **limb ($L$):** It is a jump boundary at which the normal of one of its adjacent surface is perpendicular to the viewing direction.
4. **jump ($J$):** This is a depth discontinuity which is not a limb.

### 2.2.2 Occlusion and connectivity

Surfaces are only parts of solid objects. In a segmented scene, the types of adjacency between two patches convey strong information regarding whether or not these two patches belong to the same object. Based on our labels, the adjacency information we can derive is that of *occlusion* or *connectivity*.

Let $S_i$ and $S_j$ be two surface patches sharing a common boundary $b_k$ where $k$ is the label associated with that boundary

1. $S_i$ **occludes** $S_j$ if $k$ is either a jump or a limb, and $S_i$ is closer to the viewer than $S_j$ (in a neighborhood arround $b_k$).
2. $S_i$ and $S_j$ are **connected** if $k$ is a crease, convex or concave. Furthermore, if $k$ is a convex crease, then we can conclude that both surface patches belong to the same object. If $k$ is a concave crease, then either they belong to the same object, or they belong to two different objects touching each other.

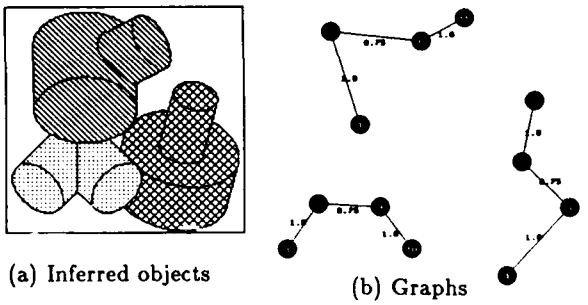
### 2.2.3 Inferring and describing objects

The match primitives are composed of graphs and subgraphs. We create a *node* for each surface patch, which contains the *unary* information about the surface patch such as the shape, orientation, and location. Then for each pair of nodes that share a common boundary, we create a *link* between them. This link contains the *binary* information between these two nodes such as the boundaries and the *possibility* that these two nodes (which represent two surface patches) belong to the same object. Thus, at the very beginning, we have a graph for each scene. Note that one node contains one surface patch only, but one link may include multiple boundaries. In the following, the terms *nodes*, *surface patches*, and *surfaces* are interchangeably used.

From the type of adjacency relationships described above, it is possible to generate hypotheses about objects. We do this by looking at each triplet ($S_i$, $S_j$, $b_k$) in turn. Whenever $k$ is a convex crease, we directly conclude that $S_i$ and $S_j$ belong to the same object. Such a strong conclusion, however, cannot be made about the other junction types. We have chosen to compute the possibility $p$ that two patches belong to the same object. This number $p$, between 0 and 1, encodes our heuristic belief that two patches belong to the same object. The following are the rules governing the way to assign and aggregate the connection possibilities $p$:

1. If there exists a convex crease (+) between two surface patches, these two surface patches must belong to the same object, and $p = 1$, otherwise,
2. If there exists a concave crease (−) between two surface patches, it is strongly believed that these two surface patches belong to the same object; however, it is also possible for the concave crease to be generated by two objects adjacent to each other, $p = 0.75$, otherwise,
3. If there exists a limb ($L$) or a jump ($J$) between two surface patches, there is no direct clue indicating whether or not they are from the same object. It may occur by either self occlusion of one object or mutual occlusion between objects. In this case, $p$ is assigned a value between 0 and 0.5 inversely proportional to the distance between the nodes (surface patches) around the limb or jump. If more than one limb or jump exists, the maximum value of $p$ is chosen. Note that we choose 0.5 as the maximum value for $p$, as opposed to the case of a concave crease, because we believe the latter gives a stronger clue.

Finally, those links with $p$ less than some threshold (here we choose 0.30) are removed, which means the two nodes are not



(a) Inferred objects (b) Graphs

Figure 2: Objects and graphs of a scene.

likely belonging to the same object. Thus, we obtain a partition of the original graph into a set of subgraphs with no links between them, each subgraph representing one (partial) object. Figure 2(a) shows the objects inferred from the scene in figure 1 in which different objects are represented by different textures, and Figure 2(b) shows the graph where circles represent the nodes whose numbers refer to the patch numbers shown in figure 1(c), and lines represent links with a value corresponding to the connection possibility $p$. It should be noted here that the grouping may not be perfect, thus merging or splitting of subgraphs may prove necessary later.

## 3 Matching Symbolic Descriptions

The system matches two scenes $A$ and $B$ at the *inferred object* level. Each object consists of a set of grouped nodes (surface patches) and links (relationships). The purpose of the matching process is to find correspondences of the objects between $A$ and $B$. At first, for each object $a$ in $A$, the objects $b$ in $B$ are ordered. Then for each $a$ in $A$, the system selects $b$ in $B$ in turn until a satisfactory correspondence is established. A *best-first* search is used to build the correspondences of objects using three types of constraints. Since the object inference may not be perfect, multiple correspondences are allowed. Finally, objects are merged and/or split to improve the correspondences. Figure 3 shows a brief block diagram of our matching method.

Our method is applicable when the difference between the viewing angles is restricted such that enough number of corresponding surface patches are visible in both scenes; however, this restriction turns out to be quite loose, as for some objects (of which we give examples later), the difference in viewing angle can be as large as 60°. We also assume that the descriptions are not dominated by viewpoint dependent components. Our technique also takes advantages of the fact that the graphs we match have a rather limited number of nodes (typically less than 30).

### 3.1 Ordering Objects

Scenes may contain a lot of objects. Searching for each pair of them to find possible correspondences would be too expensive. In this paper, we use a heuristic method to order the objects to be matched, then select object pairs based on their heuristic values to find possible correspondences between them, this process siginificantly reduces searching time.

Consider two scenes, represented by the graphs $M$ and $S$, to be

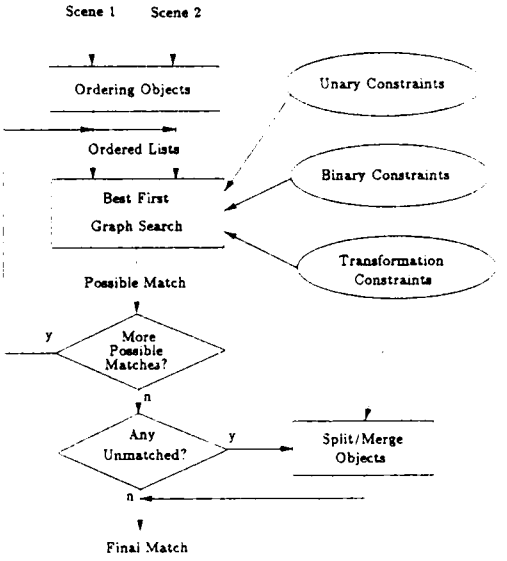



Figure 3: Block diagram of the matching process.

matched. Each graph is partitioned into the sets $M = \{M_1, \ldots, M_n\}$ and $S = \{S_1, \ldots, S_m\}$ of subgraphs, each of them representing a possible object. The matching process consists of pairing each subgraph $M_i$ of $M$ with at most one subgraph $S_j$ of $S$. In order to prune the tree to avoid searching for all possible pairs, for each subgraph $M_i$, we first order the set $S$ according to the following similarities in $M_i$ to create the list $(S_{i_1}, \ldots, S_{i_m})$:

- Number of nodes,
- Number of planar/curved nodes (surface patches),
- The visible 3-D area of the largest node.

Then the pairs $(M_i, S_j)$ are evaluated in the order of the list. If one such match, say $(M_i, S_{i_k})$ is "good enough", we do not evaluate the remaining pairs $(M_i, S_{i_l})$ for $l > k$. We therefore evaluate all pairs only in the event that there exists no good match for $S_i$.

### 3.2 Graph Searching

We use a *best-first* graph search to build the correspondences. Even though the number of nodes is assumed to be relatively small, a blind search is still prohibitive, for example, it may require as many as 10! tries for a graph with only 10 nodes. To restrict the search space, we apply a *unary constraint* (or *node-to-node* constraint) to each matching pair of nodes. If the constraint is not satisfied, the pairing of these two nodes is discarded, otherwise, a measure based on the similarity and area of these two nodes is computed. The best-first search is then applied, starting from the pair with the highest measure. Here we use a two-stage process. In the first stage, the depth of the search is retricted to 4. Only part of the nodes (with the highest measures) in $M_i$ and $S_j$ are selected and matched. These nodes are referred to as *kernel* nodes. Then in the second stage, remaining correspondences are built around these kernel nodes. In both stages, the search starts by selecting a pair of unmatched nodes $m$ and $s$, using

best-first search, where $m$ is in $M_i$ and $s$ in $S_j$. Then for each matched pair $< m_i, s_i >$, a *binary constraint* (or *link-to-link* constraint) is checked against the pair $< m, s >$. If this is satisfied, then the best transformation between the matched nodes in $M_i$ and $S_j$ is computed. The error in the estimated transformation should be lower than a specified threshold, this process is referenced as *transformation* constraint. It is important to note that the constraint imposed by the transformation is extremely powerful and supersedes all the others, more heuristic ones. In fact, it even allows us to correctly identify correspondences ignored previously because a patch was too small or mostly occluded.

The matching process continues until the above constraints can no longer be satisfied for unmatched nodes. That is, we cannot add any pair $< m, s >$ consistent with the transformation computed for all the pairs in the path from the root to the leaf. Then at each leaf of the search tree, a *match measure* based on all the matched pairs in the path to the leaf is computed. If it is good enough, the match is accepted and the search terminates, otherwise, the search continues. If all candidates are evaluated without success, then the match with the best measure is selected.

It should be noted that in selecting a pair of unmatched nodes, we do not require them to be adjacent to any of the already matched nodes or the kernel nodes. This significantly speeds up the search process as the *better* nodes are picked up first, even if they are distant from the already matched nodes.

Our method differs from the one used in [17] in the following senses:

- We use different descriptions, most of them are viewer dependent, for example, we use 3-D area instead of 2-D.
- We use transformation constraints.
- We allow large occlusion on both planar and curved surfaces while in [17] only limited occlusion is allowed on planar surfaces, and no occlusion is allowed for curved surfaces.

The three constraints and the match measure are discussed in detail below.

### 3.2.1 Unary Constraints

We first establish a node-to-node correspondence: To compare two subgraphs, we compute a set of descriptors for each node (which is a surface patch) and define a metric to compare variations, we then evaluate each pair of nodes $(m, s)$ according to this metric.

The descriptions are as follows:

- Visible area
  the 3-dimensional area that is visible to the viewer.
- Orientation
  - For planar surfaces: the direction of the normal.
  - For cylindrical or conic surfaces: the direction of the axis.
  - For other surfaces: the direction of the *least curvature*, which is defined as follows: Let $\kappa_\theta(p)$ denote the curvature of a point $p$ in direction $\theta$, and let $\kappa_\theta = \sum_{p \in P} |\kappa_\theta(p)|$, where $P$ is the patch. Then the orientation is chosen as the direction $\theta$ for which $\kappa_\theta$ is the smallest. That is, we choose the direction along which the patch is the least curved.
- Average principal curvatures $\kappa_1$ and $\kappa_2$.
  - For planar surfaces: $\kappa_1 = \kappa_2 = 0$.
  - For other surfaces: Let $O$ represent the orientation of the surface, and $V_1$ the projection of $O$ on x-y-plane. Let $V_2$ be the vector on x-y-plane perpendicular to $V_1$. Then $\kappa_1$ is the average curvature at every point of the surface patch along the direction of $V_1$ and $\kappa_2$ that of $V_2$.

  Hence $\kappa_1$ and $\kappa_2$ reflect the *flatness* of the patch.
- Estimated Ratio of Occlusion:
  A surface patch $P$ is said to be occluded by another surface patch $Q$ if there exists a limb or jump $c$ between $P$ and $Q$ where the depth value of $P$ around $c$ is less that of $Q$ (Note that the depth is encoded such that the higher the value, the closer is the point to the viewer). Let $C$ denote the set of boundaries of $P$ that are occluded by other patches, and $B$ the set of all the boundaries of $P$, then the estimated ratio of occlusion of $P$ is equal to the total length of $C$ divided by that of $B$.
- Centroid:
  The centroid of a patch is equal to the average coordinates of the visible surface points of this patch.

In measuring the similarity of two nodes $m$ and $s$ in $M_i$ and $S_j$, respectively, we first compute a normalized measure between 0 and 1 of the *difference* of each of the following properties:

$$P(a, b) = \frac{|a - b|}{\max(a, b)} \tag{1}$$

1. $D_{m,s}(1) = P(A_m, A_s)$, where $A_m$ and $A_s$ represent the 3-D visible area of $m$ and $s$, respectively.
2. $D_{m,s}(2) = P(K_m, K_s)$, where $K$ represents the average curvature $\kappa_1$.
3. $D_{m,s}(3) = P(\kappa_m, \kappa_s)$, where $\kappa$ represents the average curvature $\kappa_2$.

If any of the above measures is larger than 0.30, which represents 30% difference, then the two nodes are considered not consistent, otherwise, the similarity measure of the two nodes $m$ and $s$ is defined by:

$$D_{m,s} = \frac{\sum_{i=1}^{3} w_i D_{m,s}(i)}{\sum_{i=1}^{3} w_i} \tag{2}$$

where $w_i$ represents the weight for each item $D_{m,s}(i)$. In our experiments, we chose $w_2 = 2$, and $w_1 = w_3 = 1$.

Since the visible area is sensitive to occlusion, the threshold for $D_{m,s}(1)$ is allowed to be smaller than $0.3 + 0.7 \times R$ where $R$ is the maximum of the estimated ratio of occlusion of node $m$ and $s$.

### 3.2.2 Binary (link-to-link) constraints

Everytime a pair $P$, $Q$ of nodes is selected, it is compared to all the already matched pairs using a binary constraint. If this constraint is not satisfied, the chosen pair is discarded.

The following information is computed for each pair of nodes $P$ and $Q$:

- The type of adjacency: The adjacency between $P$ and $Q$

can be any combination or none of the following:

- $P$ is occluded by $Q$ at a jump or a limb,
- $Q$ is occluded by $P$ at a jump or a limb,
- $P$ and $Q$ are connected by a convex crease,
- $P$ and $Q$ are connected by a concave crease.

- The (3-D) length of the boundaries connecting them.
- The distance between centroids.
- The angle between the orientations.

In this second step, the following consistencies are checked for each matched pair $< m_i, s_i >$ and $< m_j, s_j >$:

1. **Connection consistency:**
   Let $l_1$ and $l_2$ represent the links between $m_i$ and $m_j$ and between $s_i$ and $s_j$, respectively. Then the types (i.e., limb, jump, convex, or concave) of $l_1$ and $l_2$, denoted as $t_1$ and $t_2$, respectively, are said to be compatible if and only if one of the following satisfies:

   - $t_1$ is equal to $t_2$, or
   - one of them is a jump and the other one is a convex crease (a change of view point may transform one into the other)
   - either one or both of them is NULL (the nodes are not adjacent, this is possible especially when shadows occur)

   Note that $l_1$ and $l_2$ may have multiple types, in this case, only major types (the boundary curves of this type are longer than a specified threshold) are considered.

2. **Direction consistency:**
   Let $\theta_1$ and $\theta_2$ denote the angles between the orientation of $< m_i, m_j >$ and $< s_i, s_j >$, and let $\theta = |\theta_1 - \theta_2|$, then the pair $< m_i, s_i >$ and $< m_j, s_j >$ *is said to be consistent in* direction if and only if $\theta$ is less than a fixed threshold $\theta_{tol}$. Here we choose $\theta_{tol} = 25°$.

3. **Distance consistency:**
   Let $L_1$ and $L_2$ denote the distance between the centroid of inertia of $m_i$ and $m_j$, and $s_i$ and $s_j$, respectively. Let

   $$L = \frac{|L_1 - L_2|}{\max(L_1, L_2)} \tag{3}$$

   then the pairs $< m_i, s_i >$ and $< m_j, s_j >$ are said to be consistent in distance if and only if $L$ is less than a threshold. Here we choose the threshold as 0.30.

4. **3D Geometry consistency:**
   For all the matched pairs $< m_k, s_k >$ other than $< m_i, s_i >$ and $< m_j, s_j >$, let $U_{ij}$, $V_{ij}$, $U_{ik}$ $V_{ik}$ represent the vector connecting the centroid of $m_i$ to that of $m_j$, $s_i$ to $s_j$, $m_i$ to $m_k$, and $s_i$ to $s_k$, respectively. Let $\theta_1$ and $\theta_2$ denote the angle between $U_{ij}$ and $U_{ik}$, and between $V_{ij}$ and $V_{ik}$, respectively. And let

   $$\theta = |\theta_1 - \theta_2| \tag{4}$$

   Then the three pairs $< m_i, s_i >$, $< m_j, s_j >$, and $< m_k, s_k >$ are said to be mutually consistent in geometry if and only if $\theta$ is less than a threshold. Here we choose the threshold to be 25°.

If these four consistency conditions are fulfilled, we say that $< m_i, s_i >$ and $< m_j, s_j >$ are mutually consistent. This only happens, however, when no abrupt changes occur as a result of a difference in viewing angle. To take into account such changes, we define the fifth condition:

5. **Enclosure:** To determine whether a surface patch $P$ is (*partially*) *enclosed* in 2-D by another surface patch $Q$, we start from the center point $C = (x, y)$ of $P$, where $x$ and $y$ are the first two coordinates of the centroid of $P$, and search outwardly in 8 directions, each 45° apart, if 6 or more out of the 8 searches encounter any point in surface $Q$, we say $P$ is enclosed by $Q$. With this definition, we accept two pairs even if they fail any of the consistency conditions above except for condition 1, as long as $m_i$ encloses (is enclosed by) $m_j$ *and* $s_i$ encloses (is enclosed by) $s_j$.

It should be noted that the above criteria are not perfect, especially when parts of the objects are occluded and mostly serve to prune the search tree. Thus it is necessary to use the transformation of the matched objects to further verify the match.

### 3.2.3 Transformation constraints

Computing the geometric transformation between matched objects not only tells how to bring matched objects in correspondence, but also helps to verify the matching process. If the error in the transformation for the current partial match is too large, the match should be abandoned.

Most of the researchers use iterative methods to compute the transformation between objects [9,12]. These methods suffer from two problems: they may need a lof of iterations before satisfactory results can be obtained; furthermore, the convergence may not be guaranteed unless a *good initial guess* is available; however, how to compute the good initial guess is usually unknown [12].

In this paper, a non-iterative method is introduced, in which the axis of the r[illegible]ion is first computed using the orientation of the matched nodes, then the angle of the rotation is obtained using a two-level search. Finally, the translation is computed from the centroids of matched nodes, using a least-square method. The details of the transformation computation is given later.

After the transformation of a match is computed, it can be used to refine the match. Since the link-to-link constraint is not perfect, some corresponding nodes may not have fullfilled the condition imposed and may therefore have been rejected. Using the transformation for the match allows us to rectify this situation. Let $A$ and $B$ be two matched objects and $T$ the transformation from $A$ to $B$, then for each unmatched nodes $a$ in $A$, let $a'$ be the node $a$ transformed by $T$, if there exists a *similar* node (satisfying the node-to-node constraint) $b$ in $B$ such that

- $b$ is the closest to $a'$, which means that the distance between the centroid of $a'$ and $b$, denoted as $D(a', b)$, is smaller than that of $a'$ and any other unmatched nodes in $B$.
- $a'$ and $b$ are close enough, which means that if $D(a', b)$ is smaller than the minimum of the width (the distance from the centroid to the closest boundary point) of $a$ and $b$.

then the correspondence between $a$ and $b$ is added to the match.

### 3.2.4 Match measure

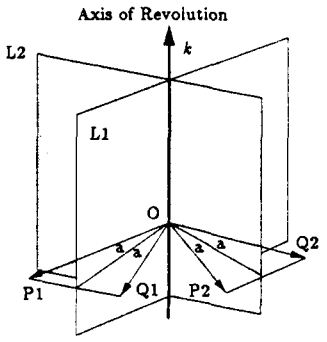



Figure 4: Computing transformations.

At the end of each path of the search tree, a match measure $H$ is computed for the current matched nodes in the path. This measure is then propagated upwards and the match with the highest $H$ is selected.

Let $T = \{< m, s > | m \in M, s \in S\}$ represent the node pairs of one possible match between objects $M$ and $S$. Let $E$ represents the transformation error of $T$, then for each pair $< m, s >$ in $T$, let $A_{m,s}$ and $D_{m,s}$ represent the minimum 3-D area and the measure of the unary constraints of $m$ and $s$, respectively. Then the match measure $H$ for $T$ is computed as follows:

$$H = \frac{\sum_{m,s \in T} A_{m,s} \times D_{m,s}}{E} \tag{5}$$

### 3.3 Computing the transformation

The rigid transformation between objects can be represented by a $4 \times 4$ matrix as follows:

$$\begin{pmatrix} R_{(3\times3)} & T_{(3\times1)} \\ 0_{(1\times3)} & 1 \end{pmatrix} \tag{6}$$

where $R$ represents the rotation and $T$ the translation, respectively. A rotation matrix can be represented by a rotation angle $\theta$ about a unit vector $k$ located at the origin. Let $k_x$, $k_y$, and $k_z$ represent the three components of axis $k$, then $R$ can be computed as follows [18]:

$$R(k,\theta) = \begin{pmatrix} k_x k_x v + c & k_y k_x v - k_z s & k_z k_x v + k_y s \\ k_x k_y v + k_z s & k_y k_y v + c & k_z k_y v - k_x s \\ k_x k_z v - k_y s & k_y k_z v + k_x s & k_z k_z v + c \end{pmatrix} \tag{7}$$

where $s = \sin\theta$, $c = \cos\theta$, and $v = 1 - c$.

To find the axis $k$, the orientation vectors for each matched node pairs are first retrieved. Let $< m_i, s_i >$, $i = 1, \ldots, N$ be the matched node pairs between two objects $A$ and $B$ where $m_i \in A$, $s_i \in B$, and $N$ is the number of matched node pairs, respectively. Let $< P_i, Q_i >$ denote the orientation vectors of the matched node pair $< m_i, s_i >$, as shown in Figure 4 for $i = 1, 2$, where both orientation vectors have been brought to the origin $O$. Assuming the rotated angle $\theta = 2a$ as indicated in Figure 4, it is easy to show that the axis of revolution $k$ for $P_1$ and $Q_1$ must lie on the plane $L1$ that equally *spaced* from $P_1$ and $Q_1$. In other words, the angle between $P_1$ and its projection on $L_1$ is equal to that between $Q_1$ and $L_1$, and this angle is equal to $a$ (unless $P_1$ and $Q_1$ coincide at $k$, in this case, however, $L_1$ is an arbitrary plane that contains $k$). The same reasoning can be applied to all the other $< P_i, Q_i >$ pairs. Thus to find the axis $k$, we only have to find all the planes $L_i$, then $k$ is at their intersections. In our implementation, we find all the intersections $k_{ij}$ of each two pairs $L_i$ and $L_j$, then for each $k_{ij}$ we compute the sum of the angle between all other $k_{ij}$, the one with the least sum is our choice of axis $k$.

The angle $\theta$ is found as follows. Let $\Theta(a, b)$ denote the angle between vector $a$ and $b$. Given a rotation $\theta$, for each matched pair $< m_i, s_i >$, $\Theta(R(\theta)P_i, Q_i)$ is computed where $R(\theta)P_i$ represent the vector of $P_i$ after rotating an angle $\theta$. Then the best $\theta$ is found by minimizing the following equation:

$$E_\theta = \frac{1}{N} \sum_i \Theta(R(\theta)P_i, Q_i) \tag{8}$$

After the rotation matrix $R$ is computed, the translation $T$ can be obtained easily. Let $< E_i, C_i >$ denote the centroids of each matched pair $< m_i, s_i >$ and $C_i' = R(\theta)E_i$. Then the translation $T$ is as follows:

$$\frac{1}{N} \sum_i C_i - C_i' \tag{9}$$

The rotation error $E_r$ and translation error $E_t$ is then defined as follows:

$$E_r = \min_\theta E_\theta \tag{10}$$

$$E_t = \frac{1}{N} \sum_i |C_i - C_i' - T| \tag{11}$$

The transformation error $E$ is given by:

$$E = aE_r + bE_t \tag{12}$$

where $a$ and $b$ are two pre-selected normalization factors. Here we choose $a = 1/360$ when $E_r$ is represented by degrees, and $b = 1/M$ where $M$ represents the size of the longest border of the scene.

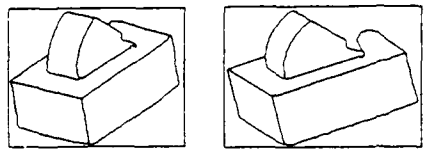
A simple example illustrates the accuracy of our transformation computation. Figure 5(a) shows two views of an object. The range images are obtained from an active range finding system built in our group [15]. The object is placed on a rotary table, and a first (left) view is acquired, the table is then rotated by precisely 25°, and a second (right) view is aquired. Figure 5(b) shows the boundaries of the object in both views, Figure 5(c) shows the boundaries of the right view after correction for the 25° rotation, and finally Figure 5(d) shows the boundaries of the right view after transformed by the *computed* transformation (the rotation angle is found to be 24°). The rotation error $E_r$ and the translation error $E_t$ in this example are 2.5° and 4.0, respectively. The image size is $237 \times 190$ and $266 \times 193$, respectively. Thus the transformation error $E$ is

$$\frac{2.5}{360} + \frac{4.0}{190} = 3.06\%$$

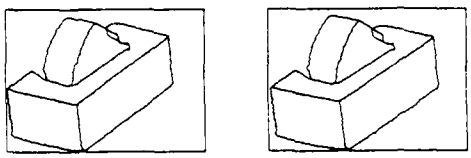
It should be noted that there are many sources which create error in our data, and contribute to the error in the transformation. One of the major sources is the inaccuracy of the calibration of the range finding system as a whole.

We have developed a method to evaluate the goodness of our transformation: For each matched pair of objects $A$ and $B$, the transformation $T$ is computed. Then object $A$ is transformed by $T$ and for each node $a$ in $A$, we compute the percentage of *over-*

(a) Original Images



(b) Boundaries



(c) The correct transform (d) The computed transform

Figure 5: The transformation computation of two views.

*lap* $R(a, b)$ between $a$ after transformation (denoted as $a'$) and the corresponding matched node $b$ in $B$. This overlap is defined as the amount of common 2-D area between $a'$ and $b$, divided by the maximum 2-D area of $a'$ and $b$. In the example we show here, the transformation error is quite comparable to the calibration error. Figure 6 illustrates the distribution of $R$ when we change the pan and tilt angles of the axis $k$ and the rotation angle $\theta$ from their computed values. Figure 6(a) shows the 3-D plot of the distribution $R$ with respective to the pan and tilt while the rotation angle is fixed at the computed value. Figure 6(b), (c), and (d) show the distribution of $R$ with respective to the pan, tilt, and the rotation angle while the other two are fixed at the computed values, respectively. The values on the x-axes of Figure 6(b), (c), and (d) show the amount of change from the computed value. As we see from these figures, the best parameters according to our measure are very close to the computed ones.

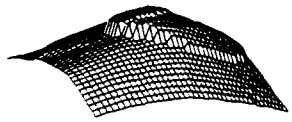
### 3.4 Some results

After the best match between two objects $M$ and $S$ is obtained, an evaluation function is applied to compute the "goodness" of the match. A match is considered good enough if at least 70% of the nodes are matched for both objects, and their total 3-D area exceeds 80% of that of the objects. Otherwise, we reject the match between $M$ and $S$.

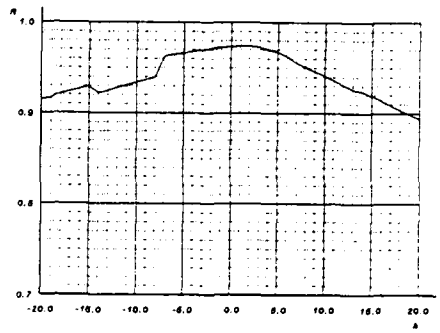
Figure 7 shows the result of matching between the two synthetic images where different textures represent individual objects and corresponding numbers indicate matched surface patches.

### 3.5 Split Objects

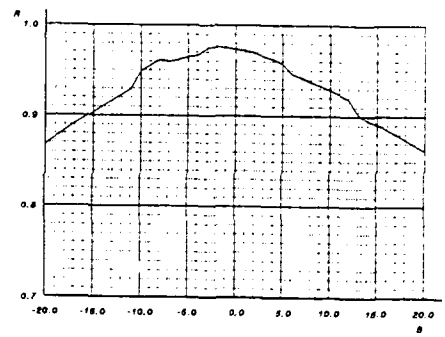
As we have mentioned before, the object inference step may not have produced perfect results: in some cases, especially when two



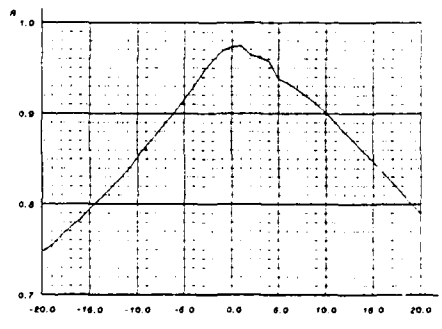
(a) Change the axis pan and tilt.




(b) Change the axis pan angle $A$.



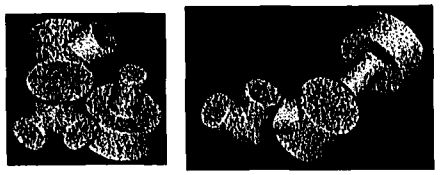

(c) Change the axis tilt angle $B$.



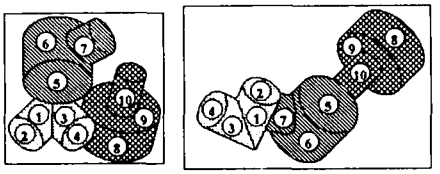

(d) Change the rotation angle $\theta$.

Figure 6: Statistics on transformation computation.

objects are touching, it is very possible that we consider them as one object instead of two. One example is shown in Figure 8 where 8(a) shows two views of a table and a chair, 8(b) shows their segmentation with the surface number for each patch. The table and the chair in the left scene are actually touching each other, i.e., surface 6 is touching surfaces 4 and 7 (and the small



(a) Original images



(b) The match

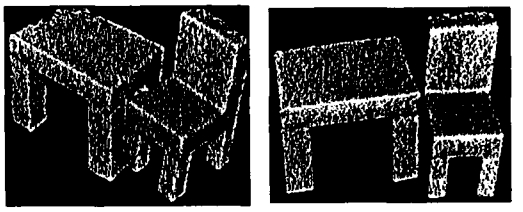
Figure 7: Match between two synthetic images.

surface 12), the object inference gives us a unique object, as shown in Figure 8(c), which is the graph of the left scene, where the numbers on the nodes represent the corresponding surface numbers. Figure 8(d) shows two possible matches of these two scenes. Without further processing, we would have to infer that there is only one object in the left scene and we have to reject one of the matches because two objects in the right scene can not simultaneously match the *same* object in the left scene.

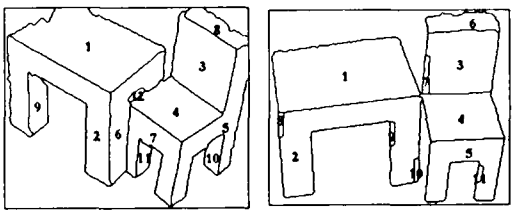
However, by examining the graphs and the matches, it is possible to *split* the two objects in the left scene. First of all, from the graph, we know that the only way to separate two objects is to cut the links between nodes (surfaces) 6-4, 6-7, and either 6-12 or 12-4, since all other links except 4-3 have a probability 1.0 which means they are not separable, and cutting 4-3 does not separate the object, thus cutting links 6-4, 6-7, 6-12, and 12-4 results in two major objects composed of surfaces 1, 2, 6, 9, and surfaces 3, 4, 5, 7, 8, 10, 11, respectively (surface 12 is too small and can be ignored). Second, by looking at the two possible matches we notice that the two *matched surface sets* on the left scene (surfaces 1 and 2 in the first match, 3, 4, 7, and 8 in the second) are disjoint, and they are contained in the first and second splitted objects, respectively. Thus it gives us a very strong clue that the split is reasonable.

From the above reasoning, we have derived a general rule to split objects, the idea being that we first split objects as much as possible, then *reconnect* those pieces already belonging to a certain match.

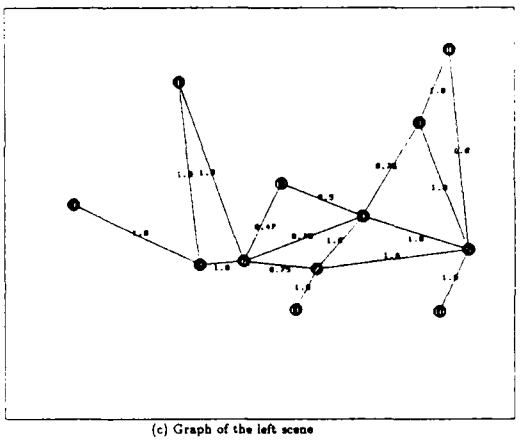
1. Find all possible matches, then independently for each of the two scenes:
2. Find all the matched surface (node) sets $M$.
3. Split the object(s) by removing *all* the links with probability less than 1.0, i.e., split them as much as possible, let $S$ denote the split objects.
4. For each pair of $s_i$ and $s_j$ in $S$, if there exists a matched node set $m_i$ in $M$ that contains some surfaces in $s_i$ and some surfaces in $s_j$, reconnect $s_i$ and $s_j$, this means $s_i$ and $s_j$ are not separable, i.e., we do not want to separate matched surfaces that are already being recognized as belonging to one object.



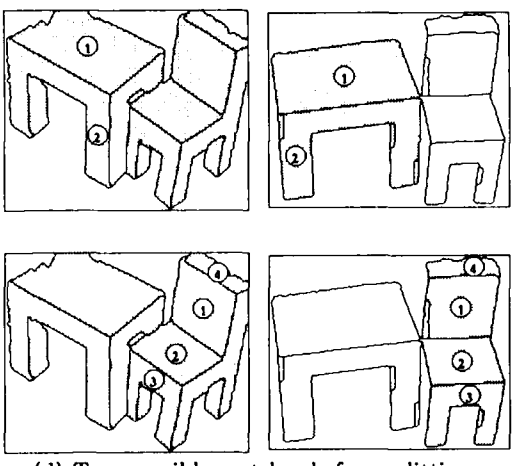
(a) Original images



(b) Segmentation



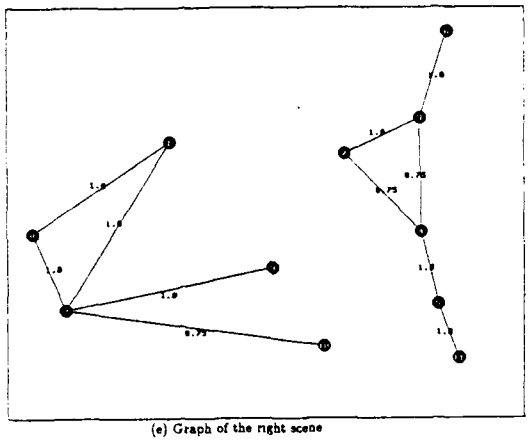
(c) Graph of the left scene



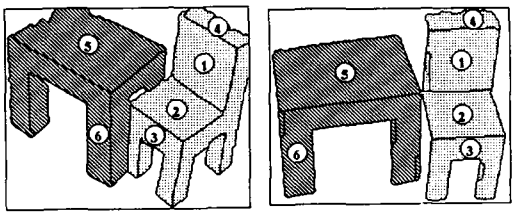
(d) Two possible matches before splitting

Figure 8: Match between furniture.

The idea in the last rule can be explained in Figure 8(e) which



(e) Graph of the right scene



(f) Final match

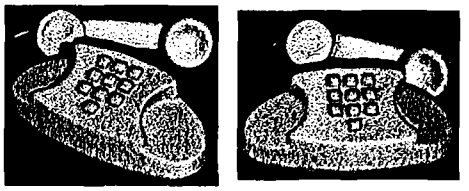
Figure 8: Match between furniture (cont'd).

shows the graph of the right scene. We do want to split the left scene by cutting the links 6-12 (or 12-4), 6-4, and 6-7; however, we don't want to cut the right scene at links 3-4 and 7-4 to separate the already matched surfaces 3, 4, 5, and 6. By the third rule above we temporarily cut the links 3-4 and 7-4 in the right scene, but we find them back by the last rule. Figure 8(f) shows the final (correct) match.

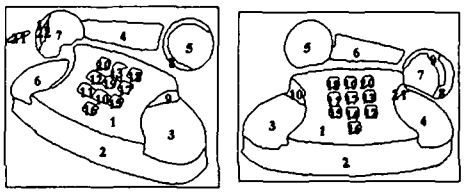
### 3.6 Merge Objects

Due to occlusions, shadows, or special view points, objects may appear as separate pieces in range images. One example is shown in Figure 9 where 9(a) shows a telephone set viewed from two different angles and 9(b) shows the segmentation. Note that the earpiece (nodes 5 and 8 in the left scene and nodes 7, 8, and 9 in the right scene) and the handle (nodes 4 and 6, respectively) are completely disjoint in both views, furthermore, the speaker (node 5) and the handle (node 6) in the right view are connected by a very large jump, therefore the object inference step failed to infer them as parts of the same object. The results of the object inference step are shown in Figure 9(c) where different textures represent different objects. Figure 9(d) shows three possible matches based on the objects found so far.

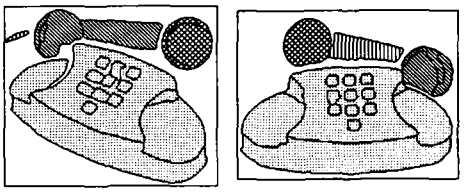
To refine the correspondence, a merging technique is implemented which brings together separate pieces that actually belong to the same objects. The idea is that by merging two sets of nodes $M_1$ and $M_2$ in the first scene to the two sets of nodes $S_1$ and $S_2$ in the second scene, not only should the *similarity constraint* (or node-to-node) between corresponding *matched* nodes in $M = M_1 \cup M_2$ and $S = S_1 \cup S_2$ be satisfied, but the *rigidity constraint* (or link-to-link) and the *transformation constraint* between each pair of matched nodes should also be satisfied. Figure 9(e) shows the final (correct) matches, including the match between the bodyset of the phone, which is the same as before merging.



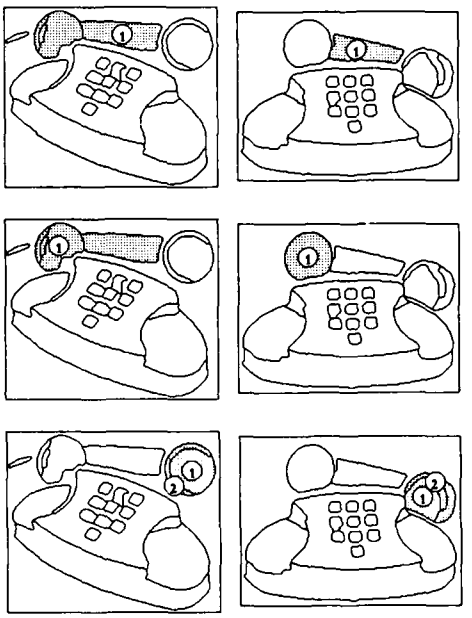
(a) Original images



(b) Segmentation

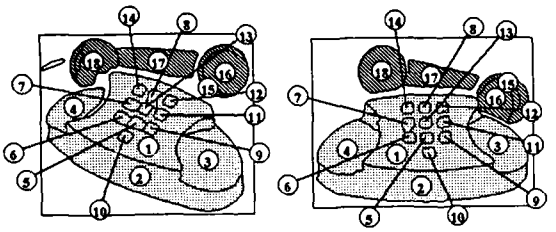


(c) Object inference



(d) Three possible matches before merging

Figure 9: Match between telephones.



(e) Final match

Figure 9: Match between telephones (cont'd).

## 4 Object Recognition

Recognizing objects is one of the major application in computer vision. In general, it requires two steps: *building models* and *recognizing scenes*. We think that a good recognition system should provide the following:

- It should automatically build object models.
- The descriptions used for models and for scenes of unknown objects should be compatible, or at least it should be easy to go from one to the other.
- The search should be efficient.

In this paper, we develope a recognition system, using the description and matching techniques mentioned before. The system automatically builds multi-view models and recognizes them in scenes of unknown objects.

### 4.1 Building models

In this paper, each object model consists of several views (6 to 8). These views are taken so that most of the significant surfaces of the model object will be contained in at least one of these views. Range images are generated for each of these views. Each image is processed and described independently and finally stored as part of the model. Several things should be noted:

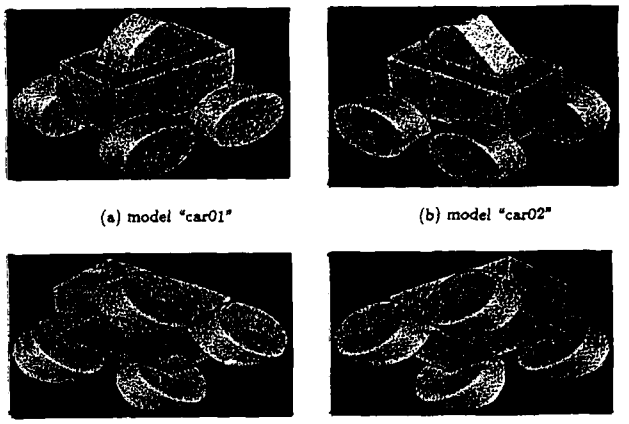
- It would be ideal if one *complete* description instead of multiple views were available for each model. For example, a description similar to boundary representation in computer graphics [20] may be a good choice; however, generating such a description is very difficult from range images. Furthermore, since scene objects are represented using *visible* surface information only, there would be inconsistency between this representation and the one used for models, thus a *translation* process would be necessary before recognition could be attemped.
- Using more views would reduce the possibility of missing significant model surfaces, for example, 60 views are used in [14] for the model of a telescope; however, this would generate an enormous search space during the recognition process. In our implementation, missing one or two significant surfaces should not affect the recognition result too much.

The multiple-view models used here have the following advantages:

- Each view can be processed and described independently,
- The description of the model views is compatible with that of scene objects.

In this paper, the following objects were chosen to build the models:

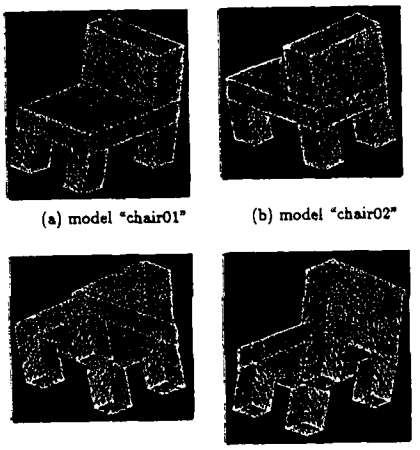
1. A toy car: which contains a lot of curved surfaces, as shown in Figure 10.
2. A chair: whose bottom view is very complicated, as shown in Figure 11.
3. A telephone: which consists of many tiny buttons that makes segmentation difficult, as shown in Figure 12.
4. A table: which contains as few as two surfaces viewed from some particular angle that would introduce ambiguities in the matching, as shown in Figure 13.



(a) model "car01" (b) model "car02"

(c) model "car03" (d) model "car04"

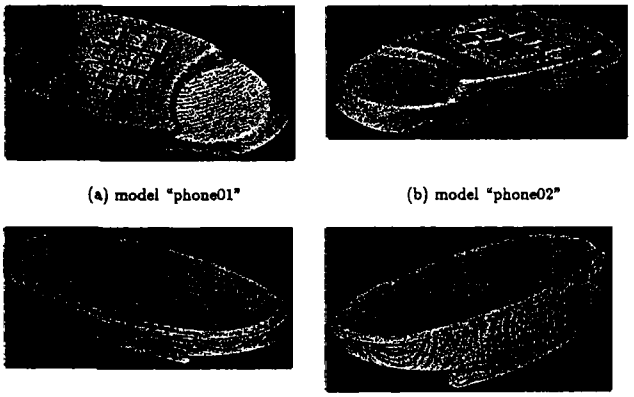
Figure 10: Four views of the model car.



(a) model "chair01" (b) model "chair02"

(c) model "chair03" (d) model "chair04"

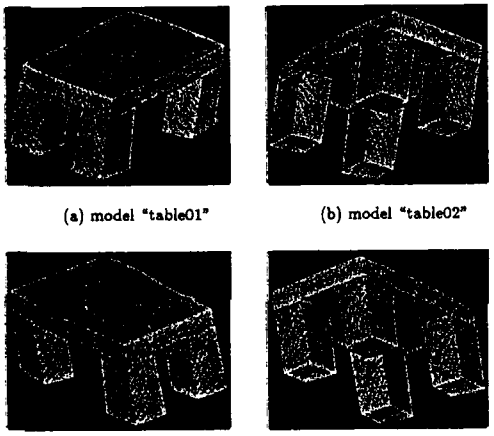
Figure 11: Four views of the model chair.



(a) model "phone01" (b) model "phone02"

(c) model "phone03" (d) model "phone04"

Figure 12: Four views of the model telephone.



(a) model "table01" (b) model "table02"

(c) model "table03" (d) model "table04"

Figure 13: Four views of the model table.

## 4.2 Recognition

The matching algorithm described in the previous section is used here to recognize scene objects. There are three steps in the recognition process:

1. **Step 1:**
   For each object, model views are ordered based on their similarities to the scene object, as described in section 3.1, then at most 5 views are selected.

2. **Step 2:**
   Graph searching is applied to find the best match between *the scene object and the preferred view. The match is* considered "good enough" if:

   (a) At least 70% of the scene nodes or model nodes are matched,

   (b) The total 3-D area of the matched nodes exceeds 80% of that of the total nodes in the scene object and the model.

   If the match is good enough, the rest of the model views are discarded, and the search is terminated. The scene object is recognized as the model object which contains the matched view. Otherwise, the ratio of the nodes ($N$) and area ($A$) are stored, and the second view is matched.

   At the end of Step 2, the view with largest $N + A$ is selected as the matched view. However, if $N + A$ is less than 1, the match is discarded.

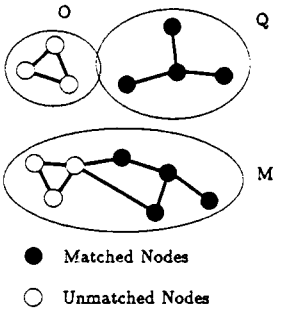



Figure 14: Merge objects.

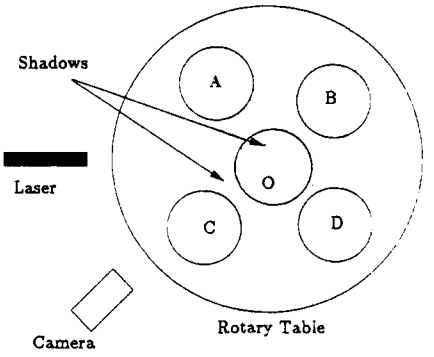



*Figure 15: Shadow problems of the rotary table*

3. **Step 3:**
   In this step, those unmatched objects are split or merged according to their number of nodes.

   - If the number of nodes is large (more than the average number of nodes of all the models), the object is split, each split object is rematched using step 1 and 2.
   - If the number of nodes for the unmatched object is small, then this object is merged to one of its adjacent matched objects. Each of the adjacent matched objects is selected one at a time. Let $O$ and $Q$ represent the unmatched scene object and the selected adjacent matched object, respectively, and let $M$ denote the matched model view for $Q$. At first, $Q$ and $O$ are merged into a new object $P$, as shown in Figure 14. Then $P$ is matched against $M$ without breaking the originally matched correspondences between $Q$ and $M$. If a new match can be found between the nodes of $O$ and unmatched nodes of $M$, and if the new match is at least as good as the original one, then the match result is updated, i.e., $Q$ and $O$ are merged and the merged object $P$ is considered to match $M$.

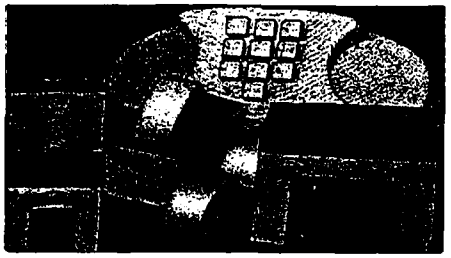


Figure 16: First scene.

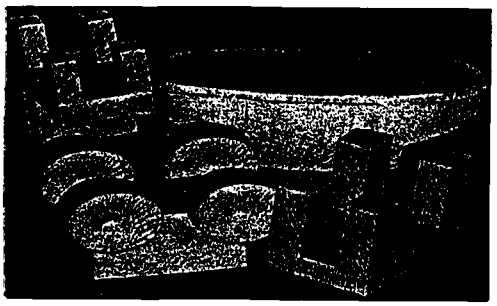


Figure 17: Second scene.

We have tested our method on two complex scenes shown in Figure 16 and 17. These scenes were obtained by first scanning each object in an arbitrary position and orientation, then generating (synthetically) the scene as it would have appeared if we had scanned it. This is necessary because of the technical difficulty in actually scanning the scene using a rotary table, as illustrated in Figure 15, which shows an object ($O$) surrounded by four other objects ($A$, $B$, $C$, and $D$) which are taller than the object $O$. As the rotary table rotates, either the laser beam or the camera is blocked by these four objects most of the time. This creates a lot of shadows that makes the surface of object $O$ almost completely missing. In order to scan these objects without casting too much shadow, we have to scan them individually.

As we may see from Figure 16 and 17 that these scenes contain a lot of occlusion and missing surface patches. The recognition results are shown in Figure 18 and 19 where (a) shows the matched model views and (b) shows the matched objects. Corresponding textures represent corresponding objects and numbers matched nodes.

Tables 1 and 2 list the statistics of the recognition. The items are interpreted as follows:

- *Object*: The object in the scene.
- *Model*: The selected model views in Step 1. The views are sorted in the order of selection.
- *Nodes Expanded*: The number of *search nodes* expanded in the search tree. It is limited to 100.
- *Max. Depth*: The maximum depth of the search tree.
- *Max. Width*: The maximum width of the search tree, it is limited to 5.
- *Decision*: The final decision of the recognition:

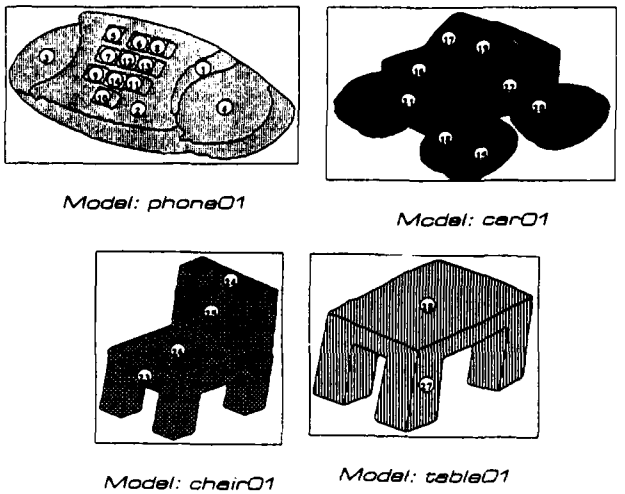



Figure 18: (a) The matched model views for the first scene.

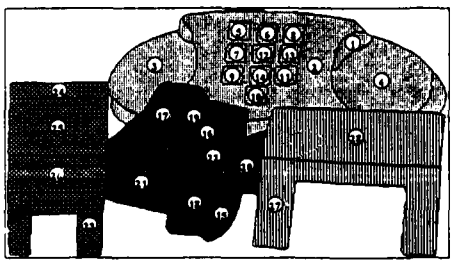
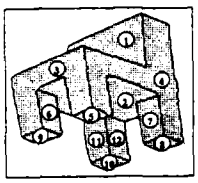


Figure 18: (b) The matched objects of the first scene.

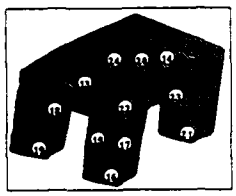
| Object | Model | Expanded Nodes | Max. Depth | Max. Width | Decision |
|---|---|---|---|---|---|
| The phone | phone01 | 20 | 10 | 4 | Matched |
| | chair03 | - | - | - | Ignored |
| | phone02 | - | - | - | Ignored |
| | chair04 | - | - | - | Ignored |
| | table02 | - | - | - | Ignored |
| The car | chair01 | 17 | 3 | 3 | Plausible |
| | car01 | 18 | 4 | 5 | Matched |
| | table02 | - | - | - | Ignored |
| The chair | chair01 | 7 | 4 | 3 | Matched |
| The table | table01 | 2 | 2 | 1 | Matched |

Table 1: The statistics for scene 1.

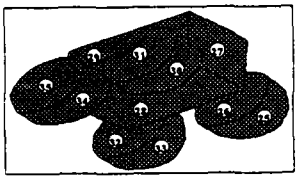
1. *Matched*: The object and the model view is considered matched (good enough in step 2). If this happens, the remaining selected model views are ignored.
2. *Ignored*: The view is ignored because a 'match' has been found.
3. *Rejected*: The match is so poor that should be rejected.
4. *Plausible*: The match between the object and the view is *fair*, which means that it is not good enough to ignore following selections, but not bad enough to be rejected. In this case, the match and its 'goodness' value (computed in Step 2) are kept for further comparison. If a true 'match' (case 1) is found during the subse-



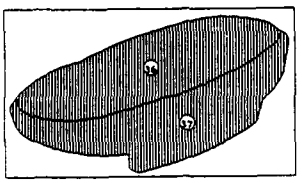
Model: chair03



Model: table02



Model: car04



Model: phone04

Figure 19: (a) The matched model views for the second scene.

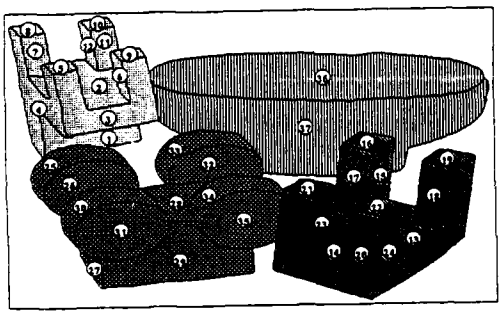


Figure 19: (b) The matched objects of the second scene.

| Object | Model | Expanded Nodes | Max. Depth | Max. Width | Decision |
|---|---|---|---|---|---|
| The chair | chair03 | 34 | 7 | 5 | Matched |
| | chair04 | - | - | - | Ignored |
| | table02 | - | - | - | Ignored |
| | phone01 | - | - | - | Ignored |
| | car04 | - | - | - | Ignored |
| The table | chair04 | 16 | 3 | 5 | Rejected |
| | table02 | 38 | 6 | 5 | Matched |
| | chair03 | - | - | - | Ignored |
| | face05 | - | - | - | Ignored |
| | car04 | - | - | - | Ignored |
| The car | car01 | 100 | 4 | 5 | Plausible |
| | chair01 | 13 | 3 | 3 | Rejected |
| | car04 | 100 | 4 | 5 | Matched |
| | table02 | - | - | - | Ignored |
| | chair03 | - | - | - | Ignored |
| The phone | phone04 | 2 | 2 | 1 | Matched |

Table 2: The statistics for scene 2.

quent exploration, this plausible match is discarded, otherwise, the plausible match with the largest value is selected as the final match.

The images are processed on a Symbolics 3640, Table 3 shows the approximate time spent for the two scenes.

Several conclusions can be drawn from the above statistics:

| Scene | time | | |
|---|---|---|---|
| | Description | Recognition | Total |
| Scene 1 | 103 min. | 15 min. | 118 min. |
| Scene 2 | 134 min. | 45 min. | 179 min. |

Table 3: Time for describing and recognizing the two scenes.

- In all the cases except the car in the second scene, the correct match is found at the first or second selected views, this proves that our heuristic indexing (Step 1) is powerful. The car in the second scene takes longer because all four model views are very similar.
- Whenever a correct view is selected, the number of expanded nodes is very small. This proves our graph searching is efficient. The largest one (100), which is the case of the car in the second scene, is due to the many similar patches which produce ambiguity.
- The objects are all segmented correctly, this proves our object inference and splitting/merging method (Step 3) are powerful.

It should be noted that if a 'good enough' match can not be found during Step 2, further analysis may be necessary to find the best match from the plausible matches. Several possibilities include increasing the resolution, focusing on one part, or changing viewing angles.

## 5 Conclusions

We have presented a matching technique and succesfully used it to recognize scenes of objects. It can also be used in other vision tasks such as acquisition of a complete 3-D object from various view-points. Our technique is not model-specific in the sense that the description and matching processes are not directly driven by any model; however, it can be used to build models and recognize objects. The main achievements of this system are summarized as follows:

- It provides a complete system to describe and match 3-D objects,
- It is data-driven so that no a priori knowledge is necessary,
- Moderately complex objects can be fully described and matched,
- Partially occluded objects can be fully described and matched,
- Information about the 3-D objects can be directly retrieved from the description,
- Objects can be inferred by grouping of surface information.

## References


[1] P. J. Besl and R. C. Jain, "Three-Dimensional Object Recognition", In *ACM Computing Surveys*, Vol. 17, No. 1, March 1985, pp. 75–145.

[2] P. J. Besl and R. C. Jain, "Segmentation Through Symbolic Surface Descriptions", In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, Miami Beach, Florida, June 22-26, 1986, pp. 77–85.

[3] M. Brady, J. Ponce, A. Yuille, and H Asada, "Describing Surfaces", In H. Hanafusa and H. Inoue, editors, *Proceedings of the 2nd International Symposium on Robotics Research*, Massachusetts Institute Technology Press, Cambridge Mass., 1985.

[4] R. C. Bolles and P. Horaud, "3DPO: A Three-Dimensional Part Orientation System," In *International Journal of Robotics Research*, Vol. 5, No. 3, pp. 3-26, Fall 1986.

[5] R. A. Brooks, "Symbolic Reasoning Among 3-D Models and 2-D Images," In *Artificial Intelligence*, 17, pp. 285-348, 1981.

[6] R. T. Chin and C. R. Dyer, "Model-Based Recognition in Robot Vision," In *ACM Computing Surveys*, Vol. 18, No. 1, pp. 67-108, March 1986.

[7] T. J. Fan, G. Medioni, and R. Nevatia, "Description of Surfaces from Range Data Using Curvature Properties," In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pp. 86-91, Miami Beach, Florida, June 22-26, 1986.

[8] T. J. Fan, G. Medioni, and R. Nevatia, "Segmented Descriptions of 3-D Surfaces," To be published in *IEEE Journal of Robotics and Automation.*

[9] O. D. Faugeras and M. Hebert, "The Representation, Recognition, and Locating of 3-D Objects," In *International Journal of Robotics Research*, Vol. 5, No. 3, pp. 27-52, Fall 1986.

[10] W. E. L. Grimson and T. Lozano-Pérez, "Model-Based Recognition and Localization from Sparse Range or Tactile Data," In *International Journal of Robotics Research*, Vol. 3, No. 3, pp. 3-35, Fall 1984.

[11] W. E. L. Grimson and T. Lozano-Pérez, "Localizing Overlapping Parts by Searching the Interpretation Tree", In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 4, July 1987.

[12] K. T. Gunnarsson, "Optimal Part Localization by Data Base Matching with Sparse Data and Dense Data", Ph.D. Dissertation, Mechanical Engineering Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania, April 27, 1987.

[13] B. K. P. Horn, "Extended Gaussian Images", In *Proceedings of the IEEE*, 72, December, 1984, pp. 1656-1678.

[14] K. Ikeuchi, "Precompiling a Geometrical Model into and Interpretation Tree for Object Recognition in Bin-picking Tasks", In *Proceedings of DARPA Image Understanding Workshop*, February 1987, pp. 321–339.

[15] J. Jezouin, P. Saint-Marc, and G. Medioni, "Building an Accurate Range Finder With Off The Shelf Components", Submitted to *Proceedings of IEEE Computer Vision and Pattern Recognition*, Ann Arbor, Michigan, June 5–9, 1988.

[16] M. Oshima and Y. Shirai, "A Scene Description Method Using Three-Dimensional Information," In *Pattern Recognition*, 9–17, 1979.

[17] M. Oshima and Y. Shirai, "Object Recognition Using Three-Dimensional Information," In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5, 4, July, 1983, pp. 353-361.

[18] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, The MIT Press, Inc., Cambridge, Massachusetts and London, England, 1984.

[19] K. Rao and R. Nevatia, "Generalized Cone Descriptions From Sparse 3-d Data", In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 256–263, Miami Beach, Florida, June 22-26 1986.

[20] A. A. Requicha and H. B. Voelcker, "Solid Modeling: A Historical Summary and Contemporary Assessment", In *IEEE Transactions on Computer Graphics and Application*, Vol. 2, No. 2, March 1982, pp. 9–24.

# SELF CALIBRATION OF MOTION AND STEREO VISION FOR MOBILE ROBOT NAVIGATION

Rodney A. Brooks, Anita M. Flynn and Thomas Marill

MIT Artificial Intelligence Lab
545 Technology Square
Cambridge, MA 02139

## ABSTRACT

We report on experiments with a mobile robot using one vision process (forward motion vision) to calibrate another (stereo vision) without resorting to any external units of measurement. Both are calibrated to a velocity dependent coordinate system which is natural to the task of obstacle avoidance. The foundations of these algorithms, in a world of perfect measurement, are quite elementary. The contribution of this work is to make them noise tolerant while remaining simple computationally. Both the algorithms and the calibration procedure are easy to implement and have shallow computational depth, making them (1) run at reasonable speed on moderate uni-processors, (2) appear practical to run continuously, maintaining an up-to-the second calibration on a mobile robot, and (3) appear to be good candidates for massively parallel implementations.

**Acknowledgments.** This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the research is provided in part by the Advanced Research Projects Agency under Office of Naval Research contract N00014-85-K-0124 and in part by the University Research Initiative under Office of Naval Research contract N00014-86-K-0685.

## 1. INTRODUCTION

This paper is about vision for a mobile robot. It is about how to build a computationally cheap robust vision system which delivers data for obstacle avoidance. The vision system is continually self calibrating, making it tolerant of normal mechanical drift. But better than that, it is tolerant of severe blows to its head like sensor platforms. After a few seconds in a trauma-induced daze it adapts to its grossly altered sensor alignments.

But wait, there's more! The algorithms require no pre-knowledge of camera focal lengths, fine orientation, or stereo baseline separation. With such quick calibration and adaption the sensors can be mounted on cheap steerable systems. We can trade cheap computation for deficiencies which arise from avoiding expensive mechanical solutions to sensor steering problems.

We begin with three observations:

- Humans are able to extract meanginful information from images without being aware of the camera geometry (e.g. baseline separation in stereo) or the focal parameters of the imaging system. They can adapt almost instantly to TV or movie images made with unknown optics and relatively quickly to disturbances in the optical pathway to their retinas. For instance a change in stereo baseline separation induced by special glasses can be adapted for in a matter of seconds. In contrast, most mobile robots today require precisely understood optics and many seconds of intense computation at the begining of an experimental run (Faugeras and Toscani (1986)) to accommodate small mechanical and electronic drifts in their systems.
- Marr (1982) points out that the *purpose of vision* depends on the task the perceiving organism is trying to achieve. Brooks (1986) demonstrates that data from a single sensor system can be used in entirely independent channels (including independent perception systems) to control different aspects of the behavior of a mobile robot. Thus there need not be just a single purpose of vision. A useful engineering consideration in building a perception system, then, is to analyze the requirements in terms of the task to be achieved using the output of the perception system.
- Much of human and animal vision is extremely fast, taking place in small fractions of a second. However, it is implemented on hardware which is extremely slow (perhaps one thousand gate delays per second) as compared with today's computer hardware (ten to one hundred million or more gate delays per second). Nonetheless biological vision is vastly superior to current computer implementations. Biological algorithms must therefore be computationally shallow in order to fit on the available hardware.

In this paper we explore some visual techniques useful for a mobile robot navigating in indoor environments. The particular task these techniques support is avoiding obstacles. Brooks (1986) shows how to separate this particular task from others that a mobile robot may concurrently be pursuing.

Noting the earlier observations, we are interested in finding *self calibrating algorithms* which are tolerant of large drifts in optical properties of the imaging system, in finding algorithms

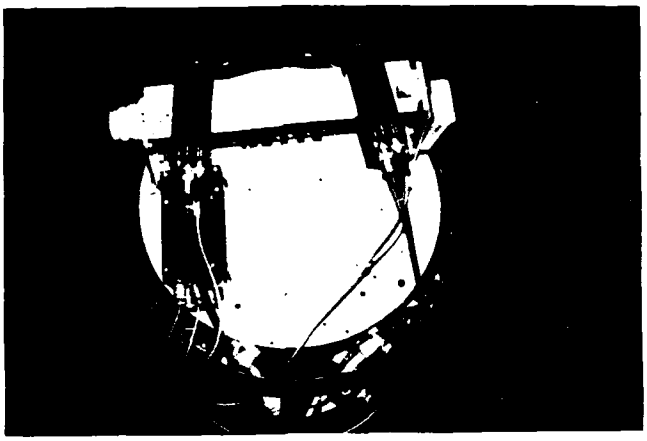


Figure 1 Two cameras are mounted on Allen using standard tripod mounts. With such mounts it is certain that the cameras will not be parallel. There is also mechanical misalignment between the camera mounts and the drive mechanism of the robot. A more expensive mechanical system and a more complex camera mounting system could overcome these limitations, but the system would still be susceptible to misalignment through minor mechanical damage. If we don't demand mechanical alignment we won't be perturbed by normal wear and tear.

which have shallow computation depth, and in finding algorithms that are well suited to the obstacle avoidance task.

The experiments we describe in this paper have been performed using Allen the robot (Brooks (1987)) as a sensor platform and an offboard Lisp machine as the computational engine. The robot has two approximately forward-looking CCD cameras. A standard camera mount is used to attach them to a tilt head, so there is considerable risk of not having the cameras pointing directly forward. Figure 1 shows the cameras mounted and on the robot.

### *1.1 Forward motion and stereo*

Our primary algorithm analyzes forward motion by tracking strong vertical edges over many consecutive images. As Bolles, Baker and Marimont (1987) have pointed out, one avoids the problem of solving for corresponding points by taking closely spaced images.

Unlike Bolles, Baker and Marimont, however, we are addressing the problem of forward rather than lateral motion, and we do not assume knowledge of camera motion or of camera angle relative to motion.

Forward motion analysis relies on straight line motion of the robot at constant, but unknown, velocity, but does not require that the camera point directly ahead, nor that it point at a known angle relative to the motion. There are strong constraints on the possible motion of features if the robot motion constraints are met, so it is often possible to detect when the motion constraints are being violated.

The particular way in which forward motion analysis supports the obstacle avoidance task is to deliver distances to obstacles in units of time to collision at the current velocity. This is the perfect coordinate system for obstacle avoidance. Steering commands and velocity change commands can be given without the need to convert from an absolute coordinate system to desired velocities for the robot.

Forward motion analysis has some surprising independence properties:

- the details of the focal geometry of a perspective camera are unimportant,
- the camera need not be pointing directly in the direction of motion (there is a simple procedure which recovers the orientation of the camera relative to the direction of motion),
- and it naturally delivers the distance to the physical artifacts giving rise to tracked features as the time to collision with that artifact in units of the inter-frame time intervals.

These properties make forward motion analysis ideal as an input to an obstacle avoidance task on the robot. Additionally it is easily implementable on hardware which supports only 16 bit arithmetic.

Unfortunately forward motion analysis has some drawbacks also:

- it doesn't deliver distances to obstacles which are straight ahead along the direction of motion (and clearly these are some of the most important obstacles to consider),
- and it is useless when the robot is stationary or turning in place. Thus when the robot stops to turn, it is completely blind in its new direction of motion until it has moved in that direction for a period of time. In a cluttered environment it may be important to be able to plan an obstacle free path before starting to move.

To compensate for these shortcomings we use a second early vision algorithm whose properties exactly complement those of forward motion analysis. We use a single scanline stereo algorithm. Stereo provides depth measurements

- straight ahead,
- and when the robot is stationary.

But stereo too has a serious drawback. Even rough depth measurements rely on a knowledge of camera focal geometry and the six parameters relating the geometries of the two cameras. Alignment errors as small as .5° for each camera can lead to wildly incorrect depth estimates, including negative depths and an order of magnitude wrong positive depths.

The main result of this paper is a simple method of continuously calibrating the stereo system to reliably deliver depths in the same units as the forward motion analysis algorithms.

### 1.2 Experimental assumptions

Man-made indoor environments give rise to images with strong vertical edges. We can make use of this fact in designing our algorithms.

The forward motion and stereo algorithms both assume one dimensional cameras with the plane defined by the image line and the optical center parallel to the ground (this was also assumed by Bolles, Baker and Marimont [1987]). In the current experiments we use regular video CCD cameras and average a swath of 16 scanlines from the middle of the image. This highlights strong vertical edges. Grey levels are taken to 8 bits and range from 0 (white) to 255 (black).

Two cameras were mounted side by side facing forward, separated by approximately 8 inches. The cameras were only approximately aligned in the forward direction. Our analysis below allows up to a $\pm 5°$ misalignment of each camera. It assumes the cameras are restricted to a field of view of 60°. Knowing the field of view lets us compute the focal ratio of the camera. The analysis assumes the two cameras are identical in this regard but is easily generalized to two different but known cameras.

In our experiments we used cameras and lens systems with approximately a 60° field of view. There is only one place, and that is in the forward motion algorithm, where knowledge of the field of view and hence focal ratio is used. In section 2 we show that in order to relate the depth estimates for stereo calibration, obtained from two cameras we must include a small error term to correct for their misalignment. It is only in this error term that the field of view is needed. We show that these errors can be at most $\pm 5\%$ so only an approximate knowledge of the field of view is necessary. For a robot which can turn in place and which has appropriate encoders to measure its turn, the camera field of view could easily be calibrated each time it turns by tracking image points over a significant portion of the image. We did not so estimate the field of view in these experiments but assumed it *a priori*.

All computation was done offboard and offline on a Symbolics lisp machine. Total computation time for the reported experiments is on the order of a few seconds running unoptimized lisp code.

We plan on porting these algorithms to a new robot with all onboard computation. On that robot we plan on using cylindrical lenses and single scan line horizontal CCD cameras. The lenses will optically average a horizontal swath — a more conventional image highlighting strong vertical edges.

For the task of obstacle avoidance we estimate that knowledge of the distance to obstacles to within $\pm 10\%$ is quite adequate.

All our experiments described in this paper give distances in terms of the robot's current speed. We did not try to measure that quantity. Our evaluation of our experiments cannot therefore be based on determinations of absolute accuracy. Rather, we compare computed relative distances to pairs of tracked objects as a measure of accuracy, as in an ideal experiment all such relative distances should remain constant. On this basis we believe we have achieved $\pm 10\%$ accuracy with very computationally simple and shallow depth algorithms.

## 2. FORWARD MOTION VISION

In this section we first derive the equations of forward motion vision without regard to sensitivity to noise and sampling errors. We then describe some practical algorithms to overcome problems due to noise and sampling errors.

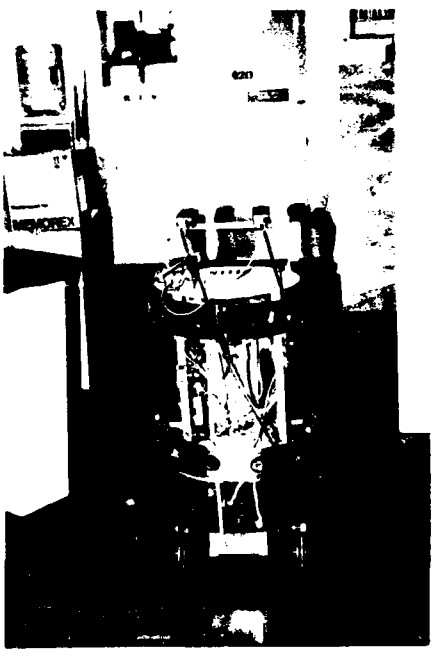


Figure 2. In our experiments Allen ran in a straight line between various obstacles. The scene was deliberately cluttered so that we could get a good calibration with only 100 stereo image pairs. We are currently limited by computer memory from collecting more than that number in real time for later analysis. In future experiments we expect to do all computation onboard and to be able to calibrate over many more images in less cluttered environments.

### 2.1 One dimensional camera geometry

Consider figure 3. It is a view from above of a camera. The *image plane* is a one dimensional strip, i.e., the camera provides only a 1-D image.

The image plane has $P$ pixels, numbered 0 through $P-1$. The *optical center* of the camera is distance $f$, measured in pixels, from the image plane. We call $f$ the *focal length*. The *optical center plane* is parallel to the image plane and runs through the optical center. A line, the *optical axis*, runs through the optical center and intersects the image plane at right angles. The point of intersection divides the image plane in two equal parts and is called the *center of view*.

Points in the world are projected onto the image plane along lines that run through the optical center. The *field of view* of the camera is the maximum angle subtended by any two vectors

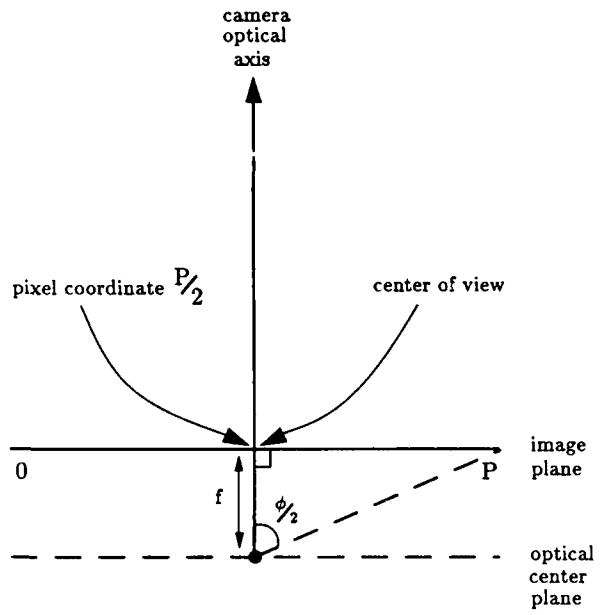



Figure 3: A one dimensional camera has an image plane $f$ pixel units from the optical center of the camera. The optical axis is perpendicular to the image plane, which is $P$ pixels wide.

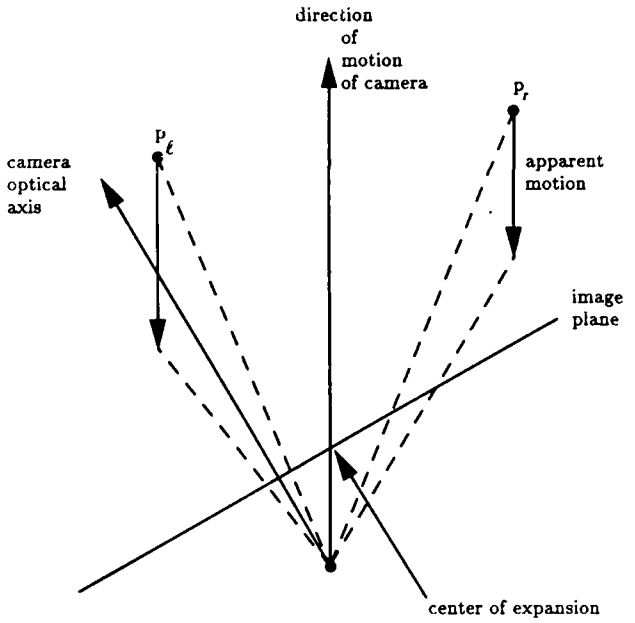



Figure 4: As a camera moves in a straight line, points to the left of the direction of motion of the camera appear to move to the left and those to the right appear to move to the right.

### 2.2 Forward motion vision geometry

Now consider figure 4. It is a view from above of a camera whose optical axis is offset somewhat from the direction of motion of the camera. The camera is undergoing pure translation in a constant *direction of motion*. The motion direction vector, the optical axis and the image strip on the image plane are all coplanar.

This model corresponds to a 1-D camera mounted parallel to the ground plane on a mobile robot which is travelling forward without turning. Typically one would mount a cylindrical lens on such a camera to enhance the strong vertical edges found in man made indoor environments. Without careful alignment of the camera it will not point directly in the direction of motion of the robot. Even if the camera is rigidly mounted on the body of the robot, the direction the robot travels with respect to its body will vary gradually over time due to tire wear and other mechanical processes. On our robot Allen we notice that the body and drive base of the robot can easily be misaligned by a few degrees. In this section we show how to calibrate dynamically for these effects with a few arithmetic operations over a few images taken as the robot is moving at constant velocity.

The trajectories of image features under the geometry of figure 4 are described by Bolles, Baker and Marimont (1987). Here we derive equations yielding the time to collision and the center of expansion under this geometry.

The image of any point $p_r$ which is to the right of the direction of motion will move rightwards in the image plane. The image of any point $p_\ell$ which is to the left of the direction of motion will move leftwards in the image plane. Points directly ahead will not move at all in the image. The projection of such points is called the *center of expansion*. We write $C_E$ for its coordinate in pixels.

In an operational robot, we envision an estimator for $C_E$ running continously and slowly and incrementally updating the estimate as it drifts. More generally what we would like to know, for the purpose of controlling a robot, is how long it will be before the robot reaches some visible point $p$. The forward motion analysis described here determines the time that will elapse for the robot to travel forward far enough that the plane, parallel to the image plane and coincident with the optical center, intersects point $p$. This assumes that the robot's velocity is constant but not that it is known.

Now consider figure 5, which illustrates the same physical setup as the last figure but with different quantities annotated. The camera has focal ratio $f$. Call the distance from the optical center to the center of expansion $f'$. The camera is misaligned by an angle $\alpha$ from pointing directly in the direction of motion. Suppose the camera is moving with constant velocity $v$ and consider what happens to the image of a point $p$. Its physical location can be described by two non-orthogonal coordinates: $x'$, its distance parallel to the image plane from the path taken by the optical center of the camera, and $z'$ its distance in the direction of motion from the optical center plane. Note that the derivative of $z'$ with respect to time, $\dot{z}'$ is the constant $-v$.

From similar triangles we can see that the distance, $r$, along the image plane, of the image of point $p$ from the center of expansion is given by:

$$r = \frac{f'x'}{z'} \qquad (2)$$

and since $z'$ is constant its derivative with respect to time is:

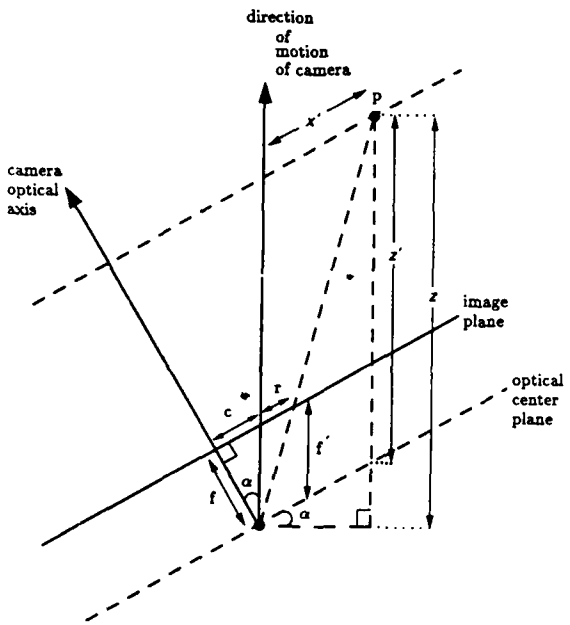



Figure 5: As a camera moves forward in some direction $\alpha$ offset from the optical center of the camera, we are interested in computing the distance $z$ to a point $p$.

$$\dot{r} = \frac{f' x'}{(z')^2}\dot{z}'.$$

We can then observe that:

$$\frac{r}{\dot{r}} = -\frac{z'}{\dot{z}'} = \frac{z'}{v}$$

The last term is the time it will take for the optical center plane to intersect $p$. Thus $r/\dot{r}$ is just the *time to collision* of the optical center plane and the physical feature whose image is being observed.

The accuracy of forward motion analysis is very sensitive to the estimate we make of the center of expansion since $r$ is measured relative to it. The position of $C_E$ can range over a third of the image when camera alignment of a 30° field of view camera is allowed to vary by $\pm 5°$. The range is not so bad for larger field of view cameras, but the quantity being measured is correspondingly smaller. Its value clearly impacts greatly the estimates of time to collision.

With perfect measurement it turns out to be easy to compute the value of $C_E$ by tracking a single point over a small time interval. Suppose we measure the image position of a point in a coordinate frame relative to the left end of the image. It has position $R$, say, in this coordinate system. Since $C_E$ should remain constant we have that

$$r = R - C_E, \quad \dot{r} = \dot{R}$$

so the time to collision for the optical center plane is

$$\frac{r}{\dot{r}} = \frac{R - C_E}{\dot{R}} \tag{3}$$

Suppose we can measure $R$ and $\dot{R}$ at two distinct times $t_0$ and $t_1$. Then the estimate to time to collision should decrease by precisely $t_1 - t_0$. Writing our estimates as functions of time we then get that:

$$\frac{R(t_0) - C_E}{\dot{R}(t_0)} - (t_1 - t_0) = \frac{R(t_1) - C_E}{\dot{R}(t_1)}$$

whence

$$C_E = \frac{\dot{R}(t_0)R(t_1) - \dot{R}(t_1)R(t_0) + \dot{R}(t_1)\dot{R}(t_0)(t_1 - t_0)}{\dot{R}(t_0) - \dot{R}(t_1)} \tag{4}$$

In summary then we can easily (given perfect measurement) estimate $C_E$ and therefore compute the time to collision of the optical center plane

$$\frac{z'}{v} = \frac{R - C_E}{\dot{R}}. \tag{5}$$

However, time to collision from the optical center plane to the point $p$, i.e., $z'/v$ is not exactly the information we want if the camera has been misaligned. In order to calibrate our stereo system in section 3 of this paper we will need depth estimates in a common coordinate system for the two cameras. Therefore we really want $z/v$ where $z$ is the component of distance from the center of the camera in the direction of motion and

$$z = z' + x' \sin\alpha \tag{6}$$

as can be seen from figure 5.

Let $c = C_E - P/2$, the distance in pixels along the image plane from the center of view to the center of expansion, then referring to figure 5

$$\sin\alpha = \frac{c}{f'}$$

so by equation (2)

$$z = z' + \frac{cx'}{f'} = z' + \frac{cx'z'}{(f')^2}$$

and since

$$f' = \sqrt{c^2 + f^2}$$

we have

$$z = z' + \frac{cx'z'}{c^2 + f^2} \tag{7}$$

Notice that since we only really know $z'/v$ (from equation (5)) this only gives us $z/v$, which in any case was what we wanted.

It is interesting to ask just how different $z$ can be from $z'$. Since $\alpha$ is restricted to $\pm 5°$, $x'$ is almost perpendicular to $z'$, so for a camera with a 60° field of view

$$\tan 30 \leq \frac{x'}{z'} \leq \tan 30$$

and since $\tan 30 - \sin 5 = 0.0503$ we see that $z = z' + x'\sin\alpha$ is within 5% of $z'$. Thus, since equation (7) is correcting such a relatively small error it is not critical to know $f$ (which can be derived from $c$ through equation (1)) particularly accurately.

We now know $z/v$, or the *time to collision* with the obstacle, very accurately using only an imprecise approximation to the

field of view of the camera as our *a priori* knowledge (which is easily gained by rotating the camera a roughly known amount). Since time to collision is precisely the right quantity to know for obstacle aviodance we don't need to try to compute the robot's velocity $v$ at all. In fact we don't even need to know how our measure of passage of time $t$ relates to external units. All we need is a steady on board clock and we can do obstacle aviodance visually.

*2.3 Tracking image features*

The above analysis suggests the idea of detecting image features and tracking them over many images.

We track edges. We simply convolve the image with a derivative of a Gaussian (the actual mask is 1, 3, 5, 9, 14, 18, 20, 18, 11, 0, −11, −18, −20, −18, −14, −9, −5, −3, −1) and then look for local maximum absolute values (Horn (1986)). We threshold the edges based on their strength of convolution with the mask. We accept only edges which have a convolution value greater than 500. We do not attempt sub-pixel localization of edges as our robot shakes enough as it rolls foward to impose at least ±1 pixel error on top of any localization noise due to discrete estimates.

Figure 8 is a data set showing edge traces from a run with the mobile robot as the robot moves forward. It is a 2-D binary array where each row is a one dimensional image and time flows downwards. Array elements are 1 (black) where an edge was detected and 0 (white) otherwise. Equation (2) and the constant velocity of the robot tell us that each edge trace is a hyperbola. If the images are taken sufficiently close together (such as in this dataset) it is very simple to track edges without having to refer to the original grey level images.

It is sufficient to buffer only two rows of the array and keep track of the direction and velocity of an edge track in order to predict a small search window and direction of search in the second row. There are two cases in searching for the next edge in a trace: at the beginning of an edge trace where the direction within the image may not be known, and later when the direction is known. At all times the possibility of noise in the edge motion must be taken into account.

When the direction is not known a default symmetric window (±3 pixels) is searched in a succeeding row. If only one edge element is found, that is taken to be the next element of the trace. If more than one edge is found the current trace is abandoned.

When the direction of motion (left or right) is known we keep track of the edge velocity at each step of the trace. The trace is predicted to move the same amount at the next step plus or minus a small window of margin in order to accomodate noise in the image and edge velocity increases; −1 pixel short of prediction through +3 pixels extra. The search proceeds in the direction of motion, and the first edge encountered is taken as the correct edge. If no edge is encountered within the window then the edge trace is abandoned. No attempt is made to hypothesize a missing edge in the given one dimensional image and to try to look in the appropriate place in the next image.

There are two ways in which the direction of edge motion can be determined.

After a few elements of a trace have been tracked the direction should be clear by comparing the position of the first and most recent pixels. If the direction is not clear then the trace can be abandoned as there is no obvious depth information in it.

However, most edge traces have only one possible direction even considering only the first row in which it appears. If $C_E$ is known then edges on the left move leftwards, and edges on the right move rightwards. Even when $C_E$ is not known, an a priori knowledge of $f$ and the maximum permissible range of $v$ determines a strip within the image where it is possible that $C_E$ might fall, and edge traces starting outside that strip have the obvious direction.

The key heuristic used by the above edge tracking algorithm to handle noise is perhaps not obvious at first glance. In fact the main idea is to abandon an edge trace when conditions get complicated. Chances are that the disturbance will last only a few images at most and the trace can be re-established as a brand new trace a few images later.

*2.4 Noise and estimation*

The analysis of section 2.1 assumes no noise and perfect measurement of various quantities. Real image sequences suffer from digitization effects and large sources of noise due to unstable camera platforms, unconstant velocity and curved rather than straight line motion. We must therefore fit all our measurements over many images.

To estimate the center of expansion we need to know $R$ and $\dot{R}$ at more than one place along an edge trace. To decrease the effects of noise it is clearly best to use more than one edge trace. We use every edge trace and continually refine our estimate of $C_E$.

Quantity $R$ is measured directly from the edge pixel array. To estimate $\dot{R}$, the edge velocity, we trade off localization in time of our estimate with accuracy of the estimate through the choice of a constant $V$. In all our experiments reported here we have used $V = 4$. We approximate $\dot{R}$ by the slope of a chord between two points on a hyperbolic trace $V$ images apart. By Rolle's theorem some point on the hyperbola between those two points has exactly that slope. We arbitrarily choose the midpoint in time. A larger $V$ reduces the magnitude effects of noise on the estimate but increases our localization error. We see that $V\dot{R}$ is simply the distance travelled by the edge trace over $V$ time intervals.

In the experiments reported in this paper we estimate $V\dot{R}$ at every pair of points on a trace separated by $3V$ steps in time, and use them to estimate $C_E$ using equation (4) (by estimating $V\dot{R}$ rather than $\dot{R}$, only one division is necessary for each estimate of $C_E$). [1] Our estimates over all time and all traces are then averaged together. We choose $3V$ so that $\dot{R}$ will have had time to change significantly and render the estimate for $C_E$ more stable. A more sophisticated algorithm could dynamically choose the step size for the comparison based on the edge velocity.

Once $C_E$ has been determined it is easy to compute the time to collision for a given edge point in a given edge using the same estimate for $\dot{R}$ as above. However since we are tracking edges and

[1] Thus each estimate relies on four points along the edge trace. Horn (1987) has suggested an iterative scheme for solving for a least squares fit along a complete trace.

Figure 6. Image strips averaged to a single scan line and ordered in time for 100 images from the left camera.

Figure 7. Image strips averaged to a single scan line and ordered in time for 100 images from the right camera.

Figure 8. The edges detected in figure 6

Figure 9. The edges detected in figure 7

since edges correspond (hopefully) to a single three space feature, the time to collision should be reducing by one every image. We can therefore smooth our estimate for the time to collision by fitting a line of slope $-1$ over $S$ images steps (i.e., $S+1$ images). A least squares fit of such a line amounts to averaging the time to collision estimate along a trace over a set of $S+1$ consecutive images.

Thus our estimates for time to collision are valid for an image $(V-S)/2$ prior to the most recently processed image.

In all our experiments reported here we have used $S=4$.

Using this algorithm the centers of expansion are estimated at pixels 243 and 274 in figures 8 and 9 respectively. These displays of edge traces are from a stereo pair of left and right cameras mounted on our mobile robot. The cameras are 576 pixels wide so these estimates say that each camera was skewed slightly to the right, with the left one skewed more than the right. This corresponds to our visual estimate from examining the camera mounts at the beginning of the experimental run. Figures 6 and 7 show the grey level images from which these edge arrays were extracted.

### *2.5 Approximation errors*

It is worth asking how well this scheme does for computing depth even with synthetic data. There are a number of sources of inaccuracies; all edge locations are approximated to an integral number of pixels, divisions (such as in equation (4)) are rounded to the nearest integer, and the edge velocity estimates are only difference approximations to derivatives. With realistic camera parameters and using $V=4$ and $S=4$ we have experimentally found accuracy on synthetic edge arrays to be about 2%.

## 3. CALIBRATING STEREO

Recall that motion vision does not get good results close to the direction of motion since estimates for $\dot{R}$ are necessarily small and therefore susceptible to noise problems. Also it is useless when the robot is still, or not moving in straight lines. Stereo vision suffers from none of these problems. Our stereo algorithm matches edges in two one dimensional images. If all the camera parameters are known in advance we could then compute depth. An accurate model of the camera geometry is not sufficient because stereo is very susceptible to small camera misalignments and so the cameras must be repeatedly calibrated.

One approach to this problem is to run a calibration procedure prior to running the robot (e.g. Faugeras and Toscani (1986)). It usually involves placing the robot in front of a known test pattern and examining the test pattern with the visual system. It is usually necessary to run the calibration procedure before every run of the robot because of mechanical (thermal and other) drift in the relative position of the optics and the robot's drive mechanism.

A second approach is to calibrate from actual images during a run. Longuet-Higgins (1981) shows how to recover all camera parameters save a scale factor from a single pair of images. However we also need to know the parameters of the cameras relative to the motion of the robot, as in general the camera platform can mechanically drift relative to the drive platform. Therefore we use the results of the forward motion algorithms to calibrate our cameras.

### *3.1 The algorithm*

We use the same edge operator as for forward motion and apply the operator to each of the left and right images.

Matching of edges in the two images is accomplished by means of a dynamic programming algorithm similar to that developed by Ohta and Kanade (1986) for two dimensional images and by Serey and Matthies (1987) for single scanline images, except that the cost functions here are different.

The dynamic programming algorithm searches for a minimum cost path left to right across the two images where there is a cost associated with matching two edges, and a cost with skipping an edge whether it is due to noise, occlusion, uncovering or something else.

The skipping cost is a constant (2000 in the experiments reported in this paper).

The matching cost is a larger number (effectively $\infty$) unless the signs of the gradients of the two edges are the same (i.e., unless the grey levels of both edges go from dark to light or vice versa). Otherwise the matching cost is the sum of squares of differences of pixel grey levels on a small window around the edge (we used 7 pixels centered on the edge, in the experiments reported here).

*3.2 What needs to be calibrated*

There are a large number of parameters needed to describe the optical system used for stereo vision. Each of the two cameras has a focal ratio, and each has six positional and rotational degrees of freedom relative to the robot. We are forced to calibrate for some of these parameters, we can ignore some because their effects are too small to notice, and others we take care of by our choice of stereo algorithm, camera design, and methodology.

Figure 10 illustrates the camera geometry we are assuming. The robot coordinate system has the $z$-axis pointing in the direction of travel of the robot. The $y$-axis is vertical and the $x$ axis is perpendicular to the direction of motion.

We assume that the two cameras have the same focal ratio $f$ or field of view.[2] Our forward motion algorithm assumed some *a priori* knowledge of this parameter and we suggested a simple way to determine a rough estimate for it. Rather than rely on such an estimate we calibrate the stereo system assuming no previous knowledge of $f$.

Now consider the geometric degrees of freedom and refer to figure 10 for definitions of geometric quantities.

- $x$ Our calibration assumes the base line separation of the cameras $B$ in the $x$ direction is unknown.
- $y$ In general one would assume this parameter to be small, but in any case our use of the average of 16 lines of grey level data in the current experiments, and our intention to use cylindrical lenses mean that larger values of $y$ will not effect the stereo measurements at all.
- $z$ There may be small errors in mounting the cameras which lead to a non-zero $z_0$. We show below that for small $z_0$ the effects are minimal and can be ignored.

*pitch* Again the averaging of scanlines or the use of cylindrical lenses takes care of this parameter.

*roll* One would expect this parameter to be small. One could measure the fuzziness of edges (since the image is really a vertically averaged, digitally or optically, image) to estimate the amount of roll present in the cameras. We will ignore this parameter.

*yaw* Each camera can swing from side to side on a standard camera mount, and in any case the orientation of the camera could easily drift mechanically from the orientation of the robot's wheels. (It certainly does on our robots.) Our calibration assumes small unknown yaw angles on each camera.

[2] It is possible to relax this assumption but the introduction of the necessary extra parameter in our calibration equations seems to make the calibration less stable. We do not have a theoretical basis for this remark – only empirical

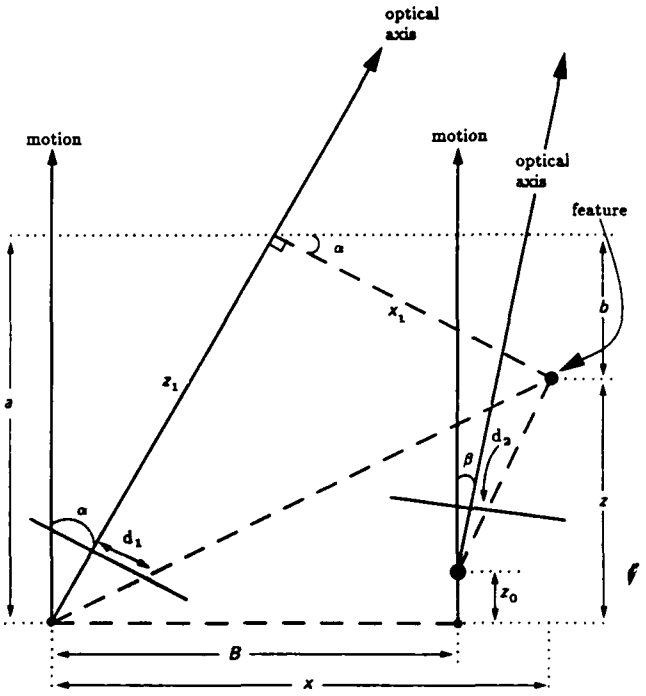



Figure 10. When there are two cameras they may both be misaligned relative to the direction of motion. They have a separation $B$ perpendicular to the direction of motion and a misalignment $z_0$ in the direction of motion. Again we are interested in the forward distance $z$ to some point.

*3.3 Calibration formulation*

Figure 10 shows a view from above of a stereo pair of cameras mounted on a mobile robot pointing approximately in the direction of motion. We assume both cameras have focal ratio $f$. The optical axis of the left camera is misaligned by angle $\alpha$ from the direction of motion, while the right camera is misaligned by angle $\beta$. The baseline separation, $B$, is measured perpendicular to the direction of motion. $z_0$ is the misalignment offset of the optical centers in the direction of motion.

We will assume that none of these five parameters (or, $\alpha$, $\beta$, $f$, $B$, or $z_0$) is known *a priori*. We will, however, assume that we know bounds on some of these quantities. We assume that $\alpha$ and $\beta$ are no bigger than $\pm 5^\circ$, that $f$, in pixels, is comparable or larger than $P$ the width of the image plane, (499 compared to 576 in our experiments) and that $z_0$ is much smaller than $B$.

The quantities we can directly measure given some matched feature in the two images are $d_1$ and $d_2$, the distances in pixels from the centers of view of the left and right cameras respectively. Due to our assumption above about the size of $f$, we know that roughly

$$d_i \ll \frac{f}{2}. \tag{8}$$

The quantity we wish to compute given some matched feature in the two images is its distance $z$ in the direction of motion. Figure 10 shows that $z$ is measured from the optical center of the left camera. The only other parameter we could compute is $x$,

the displacement of the feature perpendicular to the direction of motion. We use $x$ in deriving equations for $z$ but do not compute it explicitly in any of our experiments. Given an estimate of $f$ it can easily be recovered for use in obstacle avoidance.

Consider the quantities labelled $a$ and $b$ in the figure. They both relate to the $z$ distance of the feature from the left camera. We can write

$$z = a + b$$
$$a = z_1 \cos\alpha$$
$$b = x_1 \sin\alpha$$

where $z_1$ is the distance of the feature in the direction of the optical axis of the left camera and $x_1$ is the feature's transverse distance from that axis. Now, because we are using perspective projection we can write

$$\frac{x_1}{z_1} = \frac{d_1}{f}$$

giving us an expression for $x_1$ in terms of $z_1$ and so we can rewrite $z$ as

$$z = a + b = z_1(\cos\alpha + \frac{d_1}{f}\sin\alpha) \tag{9}$$

which is a linear expression in $z_1$. We can get a similar linear expression for $z$ in terms of $z_2$ (it involves the constant offset $z_0$) and thus get a linear equation relating $z_1$ and $z_2$.

We can do the same analysis for $x$ getting a second simultaneous linear equation relating $z_1$ and $z_2$. Solving and substituting back in equation (9) we obtain

$$z = \frac{\mu z_0 + \nu B}{\rho} \tag{10}$$

where

$$\mu = fd_2\cos\alpha\cos\beta + fd_1\sin\alpha\sin\beta + f^2\cos\alpha\sin\beta + d_1d_2\sin\alpha\cos\beta$$
$$\nu = f^2\cos\alpha\cos\beta + d_1d_2\sin\alpha\sin\beta + fd_2\cos\alpha\sin\beta + fd_1\sin\alpha\cos\beta$$
$$\rho = (f^2 + d_1d_2)\sin(\alpha + \beta) + f(d_1 + d_2)\cos(\alpha + \beta).$$

We now have an equation for $z$, the quantity we wish to compute, in terms of $d_1$ and $d_2$ the quantities we can measure. We want to derive a calibration procedure to identify the unknown parameters in this equation. We begin by noting that we can safely ignore some terms.

The sines of $\alpha$ and $\beta$ are both less than 0.1. The cosines are all greater than 0.995. Thus the dominating term in $\mu$ is roughly $fd_2$ which by equation (8) is less than half the dominating term $f^2$ in $\nu$. Since we also are assuming that $z_0$ is much smaller then $B$ we will simply ignore the $\mu$ term. We can also ignore all but the $f^2$ term in $\nu$ as the other three are each at least 20 times smaller.

Turning attention to $\rho$ we first observe that $(d_1 + d_2)$ can be arbitrarily small, and that $\sin(\alpha + \beta)$ can be larger than 0.17. Thus since either term can dominate both terms are important. The only remaining question is whether in the left hand term we can ignore $d_1d_2$ as it is never more than 1/4 the size of $f^2$. We choose to ignore it for now in our derivation of a calibration procedure, but later we compare its inclusion and exclusion experimentally and conclude that it is insignificant.

With these approximations, and dividing both the top and bottom of equation (10) through by $f\cos(\alpha + \beta)$ we get

$$z = \frac{\Lambda}{\Gamma + d_1 + d_2} \tag{11}$$

where $\Lambda$ and $\Gamma$ are constants to be determined by calibration.

Suppose we have a collection of stereo images of features with known distance $z_i$, i.e., we have a collection of $n$ triples $(d_{1i}, d_{2i}, z_i)$. For a given $\Lambda$ and $\Gamma$, we could write as a measure of closeness of fit for each triple

$$\frac{\Lambda}{z_i} - \Gamma - d_{1i} - d_{2i}. \tag{12}$$

This is both linear in $\Lambda$ and $\Gamma$ and a good measure of error in the image plane.

When we minimize the sum of squares of measure (12) for $n$ calibration triples we obtain

$$\Lambda = \frac{n\sum(d_{1i} + d_{2i})z_i - \sum(1/z_i)\sum(d_{1i} + d_{2i})}{n\sum(1/z_i^2) - (\sum 1/z_i)^2}$$
$$\Gamma = \frac{\sum(1/z_i)\sum(d_{1i} + d_{2i})z_i - \sum(1/z_i^2)(\sum d_{1i} + d_{2i})}{n\sum(1/z_i^2) - (\sum 1/z_i)^2}$$

where all the sums range over the $n$ values of $i$.

### 3.4 Calibration procedure

We collect triples to use for calibrating the stereo system by using forward motion analysis through equation (7) to get depth estimates for points visible in both the left camera and right camera. For each pair of images we know the pixel coordinates of each point for which we have a depth estimate. We run the stereo algorithm on each pair of images and for each match it finds we check to see whether forward motion delivered a depth estimate for both the edge in the left image and the edge in the right image. If so, and if those estimates are close (within $\pm 10\%$ of their average in our experiments) we use their average as the third element of a triple which includes the left and right edge pixel coordinates.

When a few tens of triples have been collected we plug them into the least squares formulation to get estimates of $\Lambda$ and $\Gamma$.

### 3.5 Experimental results

To test the preceeding algorithms we set up our robot Allen to drive in an approximately straight line with large objects on either side of its path. Figures 11 and 12 are the left and right full frame camera views from approximately the start of its path.

The cameras were connected via cables to a Lisp machine and an image was digitized every 1/15 of a second giving a stereo pair every 2/15 of a second. The robot was moving at approximately 1.5 feet per second. Thus the contribution to $z_0$ (in figure 10) due to robot motion is approximately 1.2 inches, which is small compared to our stereo baseline $B$ of approximately 8 inches and justifies our disregard of $\mu$ in equation (10).

Figures 6 and 7 show the left and right grey level motion images collected from 100 stereo pairs of standard images. Each scanline in figures 6 and 7 corresponds to averaging 16 scanlines from the centers of the original images. Time flows down these pseudo images, and is approximately 13 seconds from top to bot-

Figure 11. The left image of a stero pair at the beginning of a straight line motion of the robot.

Figure 12. The right image of a stero pair at the beginning of a straight line motion of the robot

tom. Figures 8 and 9 show the edges extracted from the time images - recall that the edge detection is one dimensional along scanlines.

In these experiments we made two passes over the edge images. The first was to estimate the centers of expansion and the second to extract motion depth estimates and correlate those with the stereo matches in order to get triples for stereo calibration. On a robot running these algorithms continuously there would only be one pass, simultaneously making minor refinements to both the centers of expansion and stereo calibrations.

In the 100 image pairs, the algorithm found 94 estimates for $C_E$ in the left image, averaging to 243, and 110 estimates in the right image, averaging to 274. A total of 92 stereo matches were found, of which 31 were strongly consistent with motion depth estimates. These were used to calibrate the stereo system resulting in estimates of

$$A = 1958.8475, \quad \Gamma = 56.970116.$$

Figure 13 shows the original left and right averaged depth estimates from the motion algorithm for each of these 31 points, along with the estimate produced by the stereo algorithm with the derived calibration. The depth estimates are in units of time between collecting images. Notice that the difference between locations of matched edges in left and right images of particular edges is sometimes positive and sometimes negative because the cameras are not aligned to be parallel.

To test the stereo calibration we took four feature points corresponding to known objects in the scene. We estimated where the robot had been when the first images in the sequence were taken (to achieve the constant velocity constraint the robot had to be moving before we started collecting images) and measured the distances to the known points in the direction of motion. We substituted the left and right image edge coordinates into equation (12) to get time to collision estimates, and divided each of those into the known distances (measured in feet) and averaged the result to get a calibration of the stereo system in feet. The estimate is 0.1874 feet per stereo pair image step, giving the robot's velocity at 1.406 feet per second. Figure 14 shows the unsurprising results of this procedure in the top four rows of the table.

We then took the same four features and searched for them 20 images later in the image sequences. The bottom half of figure 14 shows the stereo estimates for these four points. Ideally, all the time to collision estimates should be reduced by 20. They are reduced by 18, 11, 20, and 17. Note that the third of these points appeared roughly in the centers of the two images and no time to collision estimates were produced for it by the motion algorithms. The fourth column shows the distance in feet that would have been estimated if the stereo had come up with a decrease of exactly 20 units for each of the points. The sixth column shows the estimate that was actually obtained. The relative distances between the points known a priori from measurement given in the fourth column in the upper part of the table, and the estimated distances in the sixth column of the lower part of the table are quite good and can be summarized as:

| exact | est |
|---|---|
| 5.0 | 5.6 |
| 8.6 | 7.5 |
| 2.0 | 1.3 |
| 3.6 | 4.9 |
| 7.0 | 6.9 |
| 10.6 | 8.8 |

Note that these are relative distances, some of them at large distances from the robot itself.

### *3.6 Other calibration schemes*

In experiments with synthetic data the most significant term omitted from equation (10) in equation (11) appeared to be $d_1 d_2$ in the denominator. We therefore repeated the above experiment using a calibration equation of the form

$$z = \frac{A}{\Gamma + \Omega d_1 d_2 - d_1 - d_2}. \tag{13}$$

The results were almost identical:

| $l$ | $r$ | disp | motion | stereo | $l$ | $r$ | disp | motion | stereo |
|---|---|---|---|---|---|---|---|---|---|
| 144 | 163 | 19 | 50 | 59 | 168 | 189 | 26 | 69 | 63 |
| 525 | 550 | 25 | 60 | 61 | 521 | 545 | 24 | 60 | 59 |
| 169 | 197 | 28 | 54 | 68 | 516 | 541 | 25 | 60 | 61 |
| 170 | 198 | -28 | 62 | 68 | 512 | 537 | 25 | 60 | 61 |
| 507 | 532 | 25 | 60 | 61 | 172 | 202 | 30 | 71 | 73 |
| 503 | 530 | -27 | 60 | 65 | 174 | 203 | 29 | 69 | 70 |
| 174 | 204 | -30 | 74 | 73 | 149 | 137 | 12 | 28 | 28 |
| 175 | 205 | 30 | 63 | 73 | 153 | 142 | 11 | 28 | 29 |
| 155 | 146 | 9 | 30 | 30 | 73 | 68 | 5 | 32 | 32 |
| 77 | 73 | 4 | 32 | 32 | 81 | 79 | 2 | 33 | 33 |
| 85 | 85 | 0 | 34 | 34 | 90 | 90 | 0 | 36 | 34 |
| 93 | 96 | -3 | 36 | 36 | 102 | 106 | -4 | 36 | 37 |
| 105 | 111 | -6 | 36 | 38 | 9 | 48 | 39 | 112 | 109 |
| 11 | 51 | -40 | 115 | 115 | 435 | 441 | 6 | 46 | 38 |
| 13 | 54 | -41 | 122 | 123 | 430 | 439 | 9 | 48 | 41 |
| 426 | 435 | -9 | 49 | 41 | | | | | |

Figure 13. Display of the stereo calibration and results. Left two columns are pixel coordinates of edges in the left and right images, third column is their difference, fourth column is the average depth estimate from the motion algorithms running on the left and right image sequences and the fifth column is the depth estimate produced by the stereo system calibrated with this data.

$$\Lambda = 1957.6881, \quad \Gamma = 56.864132, \quad \Omega = 2.360905 \times 10^{-6},$$

an estimate of 0.1869 feet per image step (within 0.3% of the previous estimate) or robot velocity of 1.401 feet per second, and almost identical results on the check with known distances as shown in figure 15.

These experiments convinced us that (11) is certainly sufficient. But is it necessary? To check we tried calibrating with the same data to a model of stereo that assumes the cameras are perfectly aligned, i.e., using a calibration equation of the form:

$$z = \frac{\Lambda}{d_1 - d_2}. \tag{14}$$

The results were:

$$\Lambda = -1213.2903,$$

an estimate of 0.3266 feet per image step or robot velocity of 2.450 (almost a factor of 2 off), and meaningless results on known distances as shown in 16. Clearly this naive model is not sufficient.

## 4. Conclusions

In this paper we have demonstrated a number of aspects of forward motion analysis and stereo vision.

The results of forward motion analysis are fairly noisy by the unnatural standards set by most workers in computer vision and mobile robots. In this paper we first argued that those standards are not necessary, and in fact are not met by humans, most of whom operate perfectly well as autonomous mobile agents. In fact errors of $\pm 10\%$ do not seem to large to us for reliable mobile robot obstacle avoidance.

Forward motion analysis has some wonderful independence properties and only requires a small number of 16 bit fixed precision arithmetic operations to deliver depth in a coordinate system

| $l$ | $r$ | disp | known | ttc | est |
|---|---|---|---|---|---|
| 194 | 216 | -22 | 10.5 | 56 | 10.5 |
| 314 | 347 | 33 | 15.5 | 82 | 15.4 |
| 263 | 300 | 37 | 19.1 | 98 | 18.4 |
| 398 | 414 | 16 | 8.5 | 48 | 9.0 |
| 173 | 179 | 6 | 6.7 | 38 | 7.1 |
| 336 | 364 | 28 | 11.6 | 68 | 12.7 |
| 273 | 305 | 32 | 14.6 | 78 | 14.6 |
| 485 | 479 | 6 | 5.2 | 31 | 5.8 |

Figure 14. The first four rows show features with known distances (fourth column) in feet, their depth (fifth column) in time to collision units as delivered by the calibrated stereo algorithm and their depth in feet (sixth column) by calibrating the previous two columns. In the second four rows, the same points are displayed from images 20 time units later. This time the fourth column is the predicted distance based on the sixth column above and the sixth column is the estimate from stereo.

| $l$ | $r$ | disp | known | ttc | est |
|---|---|---|---|---|---|
| 194 | 216 | 22 | 10.5 | 56 | 10.5 |
| 314 | 347 | 33 | 15.5 | 82 | 15.3 |
| 263 | 300 | 37 | 19.1 | 99 | 18.5 |
| 398 | 414 | 16 | 8.5 | 48 | 9.0 |
| 173 | 179 | 6 | 6.7 | 38 | 7.1 |
| 336 | 364 | 28 | 11.6 | 68 | 12.7 |
| 273 | 305 | 32 | 14.8 | 79 | 14.8 |
| 485 | 479 | 6 | 5.2 | 31 | 5.8 |

Figure 15. Same as for figure 14 but using a much more complex stereo calibration procedure. There is almost no difference in results.

| l | r | disp | known | ttc | est |
|---|---|---|---|---|---|
| 194 | 240 | -22 | 10.5 | 55 | 18.0 |
| 314 | 347 | 33 | 15.5 | 37 | 12.1 |
| 263 | 300 | 37 | 19.1 | 33 | 10.8 |
| 398 | 414 | 16 | 8.5 | 76 | 24.8 |
| 173 | 179 | 6 | 11.1 | 202 | 66.0 |
| 336 | 364 | 28 | 5.6 | 43 | 14.0 |
| 273 | 305 | 32 | 4.2 | 38 | 12.4 |
| 185 | 179 | 6 | 18.3 | 202 | 66.0 |

Figure 16 Same as for figure 14 but using a stereo calibration that assumes the cameras are aligned parallel to the direction of motion. The results are meaningless.

natural to the task of obstacle avoidance. At the same time as it is delivering depth estimates it can continually recalibrate itself for camera misalignment relative to the direction of motion. All of these computations could easily be transfered to a parallel network of simple processors.

Despite the local noise in each forward motion estimate we demonstrated that with just a few tens of such measurements we could calibrate a stereo camera system with grossly misaligned cameras.

What does this buy us?

It means we can have a vision system on board a robot that doesn't require a *time* consuming calibration stage at the beginning of an experimental run. Rather we can *build* power-up-and-go systems. If the sensor platform drifts mechanically from the drive alignment (as it does on many real robots) over time there is no calibration problem - the algorithms given in this paper continually adjust. If there is more severe and sudden misalignment, e.g., due to a hard collision, the robot will be disoriented for only a few seconds before it accommodates.

More than this however, the possibility of simple fast dynamic calibration to the world opens up the possibility of having cheap (and hence mechanically sloppy) steerable sensor platforms. If we can quickly and cheaply compute all we need to know about how such a platform is aligned we will not feel pressure to spend inordinate amounts of money building a precise platform.

Silicon is getting cheaper a lot quicker than precision machined parts are. We *should* search for ways, as this paper does, of trading off silicon for such precision machined parts. If the analysis and algorithms are simple we have a better chance of them being robust. Of course it is critical to back up analysis with experiment - analysis in computer vision without experiment is often a worthless intellectual pursuit.

## ACKNOWLEDGEMENTS

Claudia Smith digitally drew the figures for this paper. Chris Lindblad and Dave Clemens coaxed digital images from laser printers. Berthold K. P. Horn has provided many helpful comments on earlier drafts of this paper which have improved it greatly.

## REFERENCES

**Bolles, Baker and Marimont (1987)** "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion", Robert C. Bolles, H. Harlyn Baker and David H. Marimont, *International Journal of Computer Vision*, 1, 7-55.

**Brooks (1986)** "A Robust Layered Control System for a Mobile Robot", Rodney A. Brooks, *IEEE Journal of Robotics and Automation*, RA-2, April, 14-23.

**Brooks (1987)** "A Hardware Retargetable Distributed Layered Architecture for Mobile Robot Control", Rodney Brooks, *Proceedings IEEE Robotics and Automation, Raleigh NC*, April, 106-110.

**Longuet-Higgins (1981)** "A Computer Algorithm for Reconstructing a Scene from Two Projections", H. C. Longuet-Higgins, *Nature*, 293, 133-135.

**Faugeras and Toscani (1986)** "The Calibration Problem for Stereo", Olivier D. Faugeras and G. Toscani, *Proceedings of Computer Vision and Pattern Recognition*, Miami Beach, 15-20.

**Horn (1986)** "Robot Vision", Berthold K. P. Horn, *MIT Press, Cambridge*.

**Horn (1987)** *personal communication.*

**Marr (1982)** "Vision", David Marr, *W. H. Freeman, San Francisco.*

**Ohta and Kanade (1983)** "Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming", Yuichi Ohta and Takeo Kanade, *CMU Tech Report.*

**Serey and Matthies (1987)** "Obstacle Avoidance Using 1-D Stereo Vision", Bruno Serey and Larry Matthies, *CMU Tech Report, in preparation.*

## APPENDIX-- KLUDGE FACTORS

Almost all computer vision programs have a large number of "tweakable" parameters hidden in their code. These are usually used to implement certain English phrases used in the scientific paper which describes the work. For instance there might be a phrase like "for sufficiently close edge terminations we ...". In this paper we have tried to explicitly state all such *kludge factors* that we have used. For the reader's convenience we also collect them here and give the values we used for them throughout all experiments we have done.

The edge mask (derivative of a Gaussian) is:

1 3 5 9 14 18 20 18 14 9 4 -4 -9 -14 -18 -20 -18 -14 -9 -5 -3 -1.

The threshold strength we demand of edges is 500. The same edges are used for stereo and motion algorithms.

In the dynamic programming portion of the stereo algorithm we assign a cost of 2000 to skipped matches, and the sum of squares of pixel grey level differences in a window of 7 pixels centered on the left and right edges for accepted matches.

In linking edges in the motion algorithm we start off with a

search window of $\pm 3$ pixels and later narrow that to $-1$ to $+3$ pixels from the most recent step.

To compute the edge velocity in the motion algorithms we use the slope of a chord joining edge locations $V = 4$ images apart. In estimating $C_E$ we do this twice at $3V = 12$ images apart.

We smooth the time to collision estimates over $S = 4$ time intervals (i.e., 5 images).

In deciding on consistent depth estimates and stereo matches we demand that the left and right motion depth estimates lie within $\pm 10\%$ of their average.

# AUTONOMOUS NAVIGATION IN CROSS-COUNTRY TERRAIN*

**David M. Keirsey, David W. Payton, and J. Kenneth Rosenblatt**

**Hughes Artificial Intelligence Center**
**Calabasas, CA. 91302**

## ABSTRACT

This paper describes further progress and experimentation with an autonomous robotic vehicle in cross-country terrain. Experiments on the Autonomous Land Vehicle in natural terrain were performed. An overview of the software architecture used for this achievement is discussed; descriptions of experiments and details of planning techniques are presented. We describe experiments where the vehicle avoided known and unknown obstacles in its path.

## 1. INTRODUCTION

The first cross-country map and sensor-based autonomous operation of a robotic vehicle in natural terrain was achieved in August 1987 (Daily et.al. 1987). Further experiments during December 1987 demonstrated improved performance. The vehicle reliably avoided difficult obstacles such as bushes, gullies, rock outcrops, and steep slopes as seen in Figure 1. This success was attained by Hughes through a series of experiments performed on the Autonomous Land Vehicle (ALV) at the Martin Marietta Denver test site. This paper presents a summary of experiments, the planning architecture and techniques used in these experiments. A detailed discussion of the perception part of the system is in a companion paper in these proceedings (Daily, Harris, Reiser 1988).

Figure 1. The ALV in Cross-Country Terrain

* This research was supported in part by the Defense Advanced Projects Agency under contract DACA76-85-0017

A large amount of recent progress has been made in the area of autonomous operation on roads (Wallace et. al. 1985) (Turk et. al. 1987). However, autonomous operation in an unstructured environment such as cross-country terrain presents a significantly different set of problems than road-following. In road-following applications, knowledge and expectations of the traversable surface simplify the otherwise difficult task of processing video data to detect roads. Included among the perceptual expectations are surface continuity and fairly well constrained surface properties such as width, color, and slope. In cross-country applications, there are no comparable structural expectations, but advantages are gained through the use of significantly different range-finding sensors.

This paper will first summarize our hierarchical system architecture for autonomous vehicle navigation, then present the latest planning methods used in our December experiments. Lastly, the details of the experiments are described.

## 2. SOFTWARE ARCHITECTURE OVERVIEW

The software system that was used in the experiments was a subset of a hierarchical control system developed by Hughes (Mitchell, Payton, Keirsey 1987) (Olin et. al. 1987). This system is designed to provide real-time vehicle control while maintaining the flexibility needed for operation in realistic environments. The hierarchical structure of the system reflects the need to segregate real-time planning processes from those that must assimilate a great deal of data. The four levels defined are shown in Figure 2. Higher levels in this hierarchy generate plans on the basis of highly assimilated data, relying on the lower levels to exploit more immediate data in order to execute these plans. A failure at one of the lower levels is signalled to the next higher level, which then re-assesses the situation and adjusts accordingly. At the highest level, the mission planner is used to define mission goals and constraints. At this level, the planner may interact extensively with human mission analysts to best describe the requirements for a given scenario. The mission module for the perception system is employed by the mission planner to configure sensors and predict perception system performance for various segments of the mission. The resulting mission constraints are then passed down to the map-based planner. If, at any point during mission execution, the map-based planner should indicate an inability to satisfy mission constraints, the mission planner is invoked to re-assess its initial constraints.

The map-based planner uses digital maps to plan the best route which satisfies the mission constraints. The world model for the perception system is responsible for predicting the landmarks needed to confirm the route and for marking map locations of landmarks sighted during path execution. The resulting map-based route plan is then translated into a symbolic form and passed down to the local planner. Should the local planner be unable to execute this route plan, the map-based planner may be invoked to replan the route.

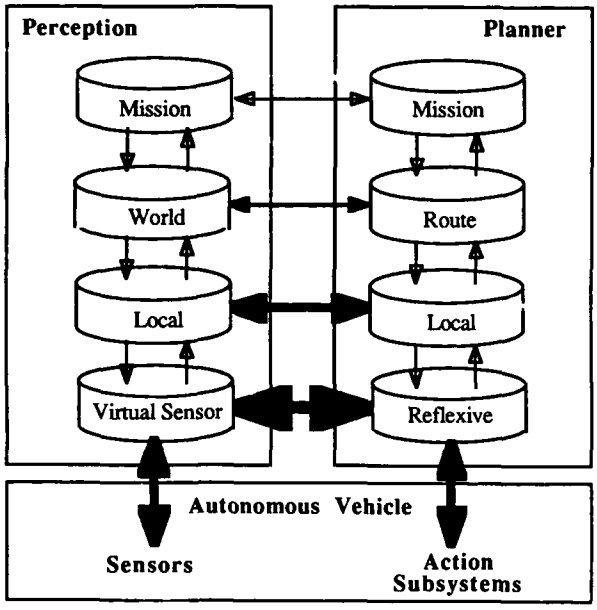



Figure 2 ALV Hierarchical Control System

The local planning module selects and monitors reflexive behavior activities in order to execute a symbolic route plan within the context of the current local environment. The corresponding local level for perception performs sensor and data fusion to accomplish object recognition and to characterize local environmental conditions. The local planning module uses this perceptual data in conjunction with map-based expectations about the local environment and knowledge of the strengths and weaknesses of various reflexive behavior activities to perform its selection task.

At the lowest level, virtual sensors and reflexive behaviors are used as the real-time operating primitives for the rest of the system (Payton 1986). Virtual sensors are sensing and processing units which can detect very specialized environmental features. At this level, knowledge assimilation is minimized in order to provide the fastest possible vehicle response. Reflexive behaviors are highly procedural units that operate on virtual sensor data to provide real-time control. In the design of reflexive behaviors and virtual sensors, various assumptions about the operating environment are made to permit faster processing. A virtual sensor is "contracted" by a behavior to provide specific information at a requested processing rate and accuracy. The virtual sensor provides the information by combining physical sensor output with appropriate processing algorithms in a manner transparent to the requestor. The behavior can then react to that information to issue speed and turn-rate commands to the vehicle.

Behaviors and virtual sensors are grouped into activities so that multiple behaviors can operate concurrently to produce control decisions. In practice, behaviors are independent processes which are compatibly configured within an activity. Activities are scheduled by the local planner as deemed necessary to achieve current goals.

No single behavior and virtual sensor combination is ever expected to be able to handle vehicle navigation problems in general; but rather, many behaviors and virtual sensors are used in conjunction, each designed to handle a specific sub-problem within the overall range of navigation tasks. In order for these specialized units to work properly, it is the responsibility of the local planner to guarantee that the selected behaviors are appropriate for the current environment. The majority of our recent experimentation with the ALV was concerned with evaluating the perception and planning systems at the level of behaviors and virtual sensors.

## 3. PLANNING

Because the primary objective of the cross-country navigation experiments was to test critical real-time perception and planning interfaces, the planning system used in the experiments consisted only of a map-based planner and a reflexive planner. The map-based planner was used to provide a description of the intended route to the reflexive planner. The reflexive planner was used to steer the vehicle in response to perceptual data received, and could be operated either with or without a route plan for guidance.

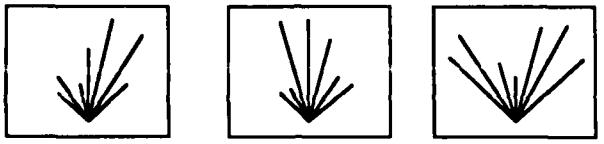
### 3.1 Map-based Planning

The map-based planner was used to decide the course that the vehicle would follow. The start and goal locations were specified by the user, and the planner used knowledge of the terrain to generate a route which best satisfied mission parameters. It then created a series of sub-goals along the path which were used to guide the reflexive planner through the area.

The digital map data available for the Martin Marietta test site was furnished by the Engineering Topographic Laboratories. The data included landcover, elevation, hydrology, roads, and landforms. The map-based planner used these maps, as well as information derived from them, to plan the best route. The planner also generated a description of regions visible from the communications tower. This visibility analysis was based on elevation, landcover, and landforms map data; it was used by the planner to ensure that the route generated met the mission constraint that line of sight with the communications tower be maintained, lest the radio link be broken. The output from the map-based planner to the reflexive planner was a series of sub-goals which described the preferred route from start to goal. The map-based planner was originally exercised on the ALV in November 1986, as described in Mitchell et. al. (1987).

### 3.2 Reflexive Planning

The reflexive planning module, guided by the map-based planner, used the output of the perception module to decide how to control the vehicle. The behaviors used in these experiments are built expressly to interface with the perception algorithms based on laser range imagery. The perception algorithms transform each range image into a Cartesian Elevation Map (CEM) (Daily, Harris, Reiser 1987). Once a CEM is produced, an algorithm which models the slope, suspension, and clearance of the vehicle is applied over this data to determine how far the vehicle may travel in various directions. The vehicle model is applied from the current location of the vehicle in the CEM reference frame forward in several directions. The resulting output is called a set of Vehicle Model Trajectories (VMT). Figure 3 illustrates typical output from this vehicle model algorithm. As shown in the figure, each scan yields a fan of potential path rays, each of varying length. Each path ray ends when the CEM data indicates either an obstacle or unknown terrain.



Scan 1
Clearing to the right

Scan 2
Clearing straight ahead

Scan 3
Forward blocked

Figure 3. Several vehicle model trajectory scans.

The reflexive planner in August was sufficiently developed to provide for the first successful cross-country run as described (Daily et. al. 1987). However, the vehicle still took some actions which did not seem to reflect intelligent decision making. One of the main reasons for this was that the vehicle model trajectory (VMT) virtual sensor specified how far the vehicle could safely travel in a certain direction, but did not indicate whether there was an obstacle detected in that direction or if we simply couldn't "see" any further. This meant that the planner had to trade off between getting close to an obstacle before judging that it had to turn and turning every time a different direction looked slightly more promising. The virtual sensor now contains information about why it is not safe to travel further than indicated, so that the vehicle can turn from an obstacle ahead of it as soon as the obstacle is detected, yet continue forward if the area further ahead is unknown. Each VMT is classified according to the reason it terminated, and can be a slope obstacle, suspension obstacle, clearance obstacle, unknown area, or any combination of the above; however, the planner currently does not distinguish between the three types of obstacles.

Another advantage of being explicitly told of known obstacles is that the behaviors can remember previously seen obstacles which are no longer apparent, for example if they are no longer within the field of view of the laser scanner. This is especially beneficial when near a gully, since gullies are difficult to detect. An obstacle is assumed to have a certain radius, and any VMT which passes through it is shortened to be no longer than the distance to the obstacle. If no VMT passes directly through an obstacle, then the VMT closest to it is treated as though it does intersect the obstacle. However, if an obstacle is at an obtuse angle from the vehicle and its distance is great enough so that it does not pose a threat, then that obstacle is ignored. When an obstacle is no longer in the area surrounding the vehicle, then the planner ceases to remember it since inaccuracies in the inertial reference system make the obstacle's relative position unreliable.

In earlier implementations, VMT's were considered in clusters, with the intent being to avoid having the vehicle try to thread its way between obstacles when other alternatives are available. The current implementation carries out this intent by considering the merits of each VMT individually, but penalizing a VMT if either of the two adjacent VMTs is not safe. Thus, a VMT is affected by its immediate neighbors but not by any other VMT. A VMT which has only one neighbor is treated as though it has another neighbor whose length is a function of the VMT and the first neighbor.

Another important change in the behaviors is the way in which a map-based goal point affects the choice of which VMT to follow. Previously, the reflexive planner would either determine that the vehicle was in a clear area and run an activity that would simply head the vehicle straight for the goal, or it would run an activity which would choose a VMT direction based on a weighting of the length of the VMT and how much that direction deviated from the heading toward the goal. In our experiments in the field, the terrain was difficult and so the planner was often running the latter activity with a predetermined fixed weight for the goal. In the current implementation, the relative importance of the goal automatically decreases as the difficulty of the terrain increases, which is desirable because as the vehicle's movement becomes more restricted it becomes more important to get clear of the rough area than to make progress directly toward the goal. Also, when the vehicle is in an obstacle-free area, the goal weight becomes predominant, so that this activity properly handles the entire spectrum of terrain difficulty making it unnecessary to switch into another activity, and thereby avoiding the overhead associated with switching.

The activity using map-based goals consisted of the following behaviors:

**Monitor-Sub-Goals:** Responsible for determining when a map-based goal is achieved and selecting the next goal. If, while attempting to satisfy the current goal, the vehicle enters an ellipse which has the current goal and the next goal as foci, then this behavior decides that it is best to go on to the next goal and bypass the current one.

**Monitor-Final-Goal:** When the last of the map-based goals has been achieved, this behavior will inform the reflexive planner that the current activity has accomplished its task.

**Seek-Best-Path-To-Goal:** Selects which VMT the vehicle should follow, taking into consideration the length of a VMT, whether it terminates at an obstacle or an unknown area, the length and classification of adjacent VMTs, previously detected obstacles, and the location of the current goal.

**Adjust-Safe-Distance:** Determines the safe distance remaining along the VMT which has been selected, decrementing this value as the vehicle moves forward.

**Adjust-Speed-For-Distance:** Sets the vehicle speed based on the safe distance remaining, at first slowing gradually, then more abruptly as the vehicle approaches an area not known to be traversable.

**Slow-For-Turn:** Determines the maximum speed at which the vehicle may safely travel based on the current turn-rate and the lengths of the VMT's between the vehicle's current position and the VMT chosen as the one to follow.

**Slow-For-Goal:** As the vehicle approaches the current goal, this behavior makes sure that, given the current turn-rate, the vehicle is not travelling so fast that it will pass by the goal without satisfying it. Without this behavior, the vehicle could pass the goal, turn back and pass it again, and continue circling the goal in this way without ever reaching it.

**Stop-And-Turn-When-Blocked:** Stops the vehicle if all non-obstacle VMTs are below a certain threshold, and rotates the vehicle a specific number of degrees in the direction which appears more promising. It will then wait to receive a set of VMT's based on a laser range scan taken from the vehicle's new heading, and if all paths are still blocked, it will rotate further in the same direction as before and repeat the process. Once a clear path is detected, this behavior will relinquish control.

The behaviors described above provide the planner with the capability to react to the nearest obstacles in a consistently reasonable manner. Thus, higher levels in the planning hierarchy can now perform more abstract reasoning with the

assurance that the reflexive planner will be able to carry out the assigned task and safely navigate the vehicle through cross-country terrain.

## 4. EXPERIMENTS

The December ALV experiments performed by Hughes were intended to improve autonomous cross-country navigation with the ALV and correct various problems that were encountered during the earlier August experiments. As a result, we were able to demonstrate significant improvements such as higher speed (up to 3 km/hour), longer distances (exceeding 500 meters), longer running times (averaging 20 minutes), and increased difficulty of terrain. In addition, improvements in our recording capabilities have allowed us to retain a wealth of information that will help us make further refinements to our planning and perception systems.

After the August experiments, we realized that various aspects of our simulation were not accurate, and that as a result, some of the reflexive behaviors for controlling the vehicle would fail in certain situations. We found that during development, we had used an incorrect model of how the vehicle was allowed to turn and how it responded to turn commands, leading to some serious control failures when the vehicle encountered situations requiring careful maneuvering. The August experiments were followed by various enhancements to our simulation and a significant re-design of many of the reflexive behaviors. After performing extensive simulations using these new behaviors, we were ready to go back to Martin Marietta with confidence that we could perform our cross-country experiments with greater reliability while also increasing the overall level of difficulty.

In our first set of experiments, we tested our new control interface by teleoperating the vehicle through a road intersection and in various patterns of circles and zig-zags on an empty flat field. Satisfied that our control methods were correct, we then performed ten runs in the flat field using traffic cones and trash cans as obstacles. These tests were primarily to isolate and verify reflexive planning obstacle avoidance and control capabilities. We performed tests both with and without goals, and found that goals were necessary to keep the vehicle from wandering off to the side of the field. With a maximum speed of 2 km per hour, and an average speed of 1 km per hour, the vehicle had a tendency to slip and get stuck in the muddy soil. Also, we found that a cement foundation with some metal spikes was not detected as an obstacle by our perception algorithm. This was no surprise, since the five-inch spikes did not violate the twelve-inch clearance threshold used in the perception algorithm, but it was seen as a problem since they could possibly have punctured a tire. Despite these problems, the vehicle traveled through the obstacles, usually meeting its goals without much difficulty. These tests convinced us that our behaviors for obstacle avoidance were suitable for medium-density obstacle fields.

After these experiments were completed, the vehicle was refueled, and moved from the flat field to a hillside. This hillside was selected because it offered many challenges which were not present in the ordinary flat field. The general slope of the hillside was close to fifteen degrees, just bordering on the specified eighteen degree maximum slope for the vehicle. The hillside also contained numerous small and mid-sized rocks, many of which would serve as natural obstacles. In addition, there were both small and large clumps of trees to be avoided. Among the more challenging obstacles were some gullies (typically three feet wide and two feet deep) which could only be crossed at a few locations, yet they were not easily seen from a distance. The most prominent feature of this hillside was a large rock outcrop which came to be known as the Eiger, and hence, the hillside itself was called the Eiger Field. To one side of the Eiger was a gully, and beyond this gully the hillside leveled off into a fairly flat area.

Our primary mission objective in the experiments that followed was to make the vehicle travel from the hillside, across the gully, into the field where we had placed some trash can and traffic cone obstacles, and then turn around and return to its original starting point. The Eiger played an important part in this scenario, because it could block radio communication between the vehicle and the computers in the lab. Because of this problem, it was necessary that the vehicle go around the Eiger on the high side, and then traverse between the Eiger and the gully until it got to a point where the gully could be crossed. A route which met these requirements was generated automatically by our map-based planner and used line-of-sight criteria to ensure reliable radio communication.

The first set of Eiger Field experiments was intended simply to repeat similar runs performed in August. The map-based plan was expressed in terms of a sequence of goal points placed approximately fifty meters apart for which goal satisfaction entailed passing within a five meter radius of each successive point. The first point in the plan was at the bottom of the hillside, near a clump of trees. The plan then specified a series of four points which curved around the high side of the Eiger. There was then a goal point near the low side of the Eiger, indicating the correct location for crossing the gully. A final point was located at the far side of the flat obstacle field. An identical set of points, listed in reverse order, was used to specify the return route.

In the first two experiments in this terrain, the vehicle navigated successfully up the hill, avoided a few obstacles, and went around the high end of the Eiger with no problem. The vehicle then proceeded back down the hill with the gully on its left and the Eiger on its right. As it approached the goal point at the desired gully crossing, the vehicle encountered a three-foot high, two-inch wide post used for marking an underground pipe. In both runs, the vehicle made a left turn to avoid the post. In the first run, the vehicle continued straight toward the gully, forcing the run to be aborted. We later found this problem to be related to an incompatibility between our setting for the clearance threshold and the cautiousness of the vehicle operators. In the second run, the vehicle passed between the post and the gully and began entering the flat area. However, it had failed to satisfy a goal located to the right of the pole and therefore attempted to turn around and head back to the missed goal. The vehicle made a series of sharp left turns, digging its left wheels into the soft soil and becoming stuck so that it was unable to complete the route.

As a result of this experiment, we realized that there was a problem with constraining the vehicle to achieve all the goals in their proper sequence, regardless of how the vehicle deviated from its intended path. In situations where the vehicle has failed to achieve its current goal but is already making progress toward the next one, the current goal should be abandoned and the vehicle allowed to pursue the next goal. We needed to modify the goal-selection algorithm so that it would detect these situations and respond accordingly. To do this, an ellipse was defined with the current and next goals as the foci; should the vehicle enter this ellipse, then the current goal would be bypassed and the next goal would become the current goal.

Using this new goal-selection algorithm in the next run, the vehicle smoothly went up and around the Eiger, but then surprised us by turning around at the gully crossing, heading back up alongside the Eiger, and back to its original starting point. After a brief analysis of this run, we realized that the vehicle skipped a portion of its route because of an error in the rule for ignoring the current goal and selecting its successor. In this case, the problem arose due to the configuration of the goals. Since the route doubled back on itself, the points in the loop were ignored. Although this problem was corrected by making the goal in the field a required goal, this confirmed our belief that symbolic route description techniques would be needed for future missions because the simple description lacks critical knowledge about the purpose and meaning of various sub-goals.

With this correction in place, we tried the run again. This time, the vehicle got around the Eiger, crossed the gully successfully, and avoided several obstacles in the flat area to reach its farthest goal. Then, the vehicle made a gradual left turn to make the return trip to the starting point. Because of obstacles in the field, the vehicle made an excessively wide turn, causing it to approach the gully at point that was intraversable. The gully was not detected as an obstacle, so the run had to be aborted. We tried this run again, this time extending the route by moving the start position. The extended route began near a road, and went down the hill and around the large clump of trees before joining with the previous route. Again, the first half of the run was completely successful, but on the return trip, the vehicle failed to avoid the gully.

Gullies are difficult terrain obstacles to detect and avoid for a number of reasons. Since they represent depressions in local terrain, at further distances and more oblique viewing angles gullies are often not detectable even to humans. The full depth (and therefore the degree of danger and classification as an obstacle) is not visible until the viewing angle and distance allow the sensor to look down to the bottom of the gully. The laser range scanner is further hampered by lower resolution at further distances and average-reflectance weighted footprints at oblique angles. The transformation to the CEM can also produce inaccuracies depending on the resolution and method of determining elevation at a given pixel. For example, a CEM with 12 inch per pixel resolution which keeps only the maximum elevation at a pixel can easily obscure a narrow gully less than 12 inches wide. To further compound the difficulty of gully detection, grass and vegetation often obscure portions of the gully, forcing confident detection to occur at dangerously close distances. For the cases mentioned above, it is also likely that the clearance threshold set for 12 inches was not safe enough for the actual vehicle clearance and conservative oversight of the operators.

Suspecting that our obstacle arrangement was preventing the vehicle from reaching the gully crossing on its return trip, we changed the obstacle layout and performed several more runs starting at the gully and heading out into the flat area. The first time we tried this, the vehicle started off to the left and then made a wide right-hand turn to head back to the gully. This time, it successfully crossed the gully, but instead of turning right to go up around the high the side of the Eiger, it turned left and went on the low side. Radio contact was not lost, but the vehicle failed to avoid a small rock so the run was aborted.

After repeated incidents of the ALV operator stopping the vehicle for rocks or gullies which were not detected as obstacles by our perception algorithm, we came to realize that the twelve inch clearance threshold we were using was not consistent with the ALV operator's clearance requirements. After discussing the problem with the operator, we came to the conclusion that our perception algorithm should use a six inch threshold to ensure vehicle safety. We were concerned about using such a small threshold because of the possibility that an excessive number of spurious obstacles would be detected, and because it was very close to the three-inch resolution limit of the data, making it highly sensitive to noise. Having no other viable alternatives, we used the six inch clearance threshold in all runs that followed.

After two more false starts with this new threshold, the final run of the day was very successful. Again, we started at the base of the gully and headed out into the field. As in the previous run, the vehicle started off to the left and then turned to the right to meet its goal. It then continued to the right, heading back to the gully crossing. In doing so, it encountered a sewer pole and rock, causing it to make a much wider turn than might otherwise have been desired. Finally, it made a sharper turn to head more directly toward the gully crossing, slipping some and digging deeply into the earth. As a precaution, the operator had to briefly stop the vehicle as it was turning to remove a dirt clod from its path. Once autonomous mode was resumed, the vehicle continued toward the gully and passed over the survey marker goal point with its left tires. The vehicle then turned right, headed up alongside the Eiger, and passed to the right of the pipe marker. It was apparent that by starting off to the left at the beginning of this run, the vehicle had a better approach angle for crossing the gully and making the right-hand turn to go around the Eiger. The vehicle turned left at the high side of the Eiger, and then headed down the hill to reach the final goal.

Having success at the gully crossing, we decided to try the extended run again. This too was a fairly successful run, starting near the road, going down the hill to get around the clump of trees, then back up and around the high side of the Eiger, crossing the gully, traversing the obstacle field, then taking a broad left-hand turn to get back to the gully, and this time it crossed the gully on the way back. Unfortunately, it failed to turn right at this point, going left behind the Eiger instead. Radio contact was poor, but sufficient to keep the experiment going. The vehicle then looped back up around the high end of the Eiger again, attempting to reach a goal that hadn't been attained yet. Finally, it got too close to the gully and had to be stopped. The failure of the right-hand turn was later traced to an error in our algorithm for translating turn-rate commands into the control trajectories required by the ALV pilot software. The fact that the vehicle went back up the hill to reach goals that were no longer relevant again illustrates the deficiency of the simple goal-point method for describing a route, but it also provides some insight into how a local planning level might intervene when the reflexive planner encounters unexpected circumstances.

A number of runs followed in which we experimented with reduced goal lists. Since we had found no real problem in maintaining radio communications as the vehicle traversed on the low side of the Eiger, we started with a very simple goal list containing the starting point up near the road, an intermediate point at the gully crossing, and then a final point out in the flat area. In the first of these runs, the vehicle started out heading straight for the gully crossing as expected, but instead of detecting the rough terrain above the large clump of trees, the vehicle headed directly into an area with numerous deep and narrow gullies. The gullies were detected after the vehicle was too close to turn away. If perception had detected the gullies sooner, or if the vehicle could have been backed up autonomously, this problem might have been overcome.

In a second try, we added another goal point near the base of the clump of trees, but again, the vehicle got stuck in the gullies near

the beginning of the run. We then moved the goal point at the base of the clump of trees in order to guide the vehicle away from the troublesome area. This time, the vehicle successfully went around the clump of trees, but as it went toward the intermediate goal at the gully crossing, it was unable to avoid a tree. As the vehicle approached the tree, it saw a clear path slightly to its right, but in attempting to make the turn, the faulty algorithm for converting turn-rates into trajectories caused it to continue straight ahead. Finally, the vehicle found itself completely blocked by the tree.

In our next experiment, we retained the reduced goal list and corrected an error which had been effectively cutting our maximum speed from 2 kilometers per hour to 1 kilometer per hour. This test constituted our first complete run from the goal near the road to the flat area and back. The higher speed, averaging about 1.8 kilometers per hour, actually seemed to result in smoother overall control. The vehicle passed below the clump of trees with no problem, and then went straight for the other tree that had previously caused problems, but this time it avoided it. The vehicle went between two rocks and then avoided the Eiger, coming fairly close, but still missing it. Radio transmissions were poor, but it got past this area and proceeded across the gully. Once in the flat area, the vehicle avoided a few trash cans and traffic cones to reach its farthest goal, then made a wide left turn to come back. Again avoiding a few obstacles, it crossed the gully somewhat higher than intended, then headed down between the Eiger and pipe-marker post. Passing through this area, the vehicle then turned right and headed toward the base of the clump of trees. Along this segment of its path, the vehicle came upon an eight inch high rock with surrounding grass of about the same height. Because there was no significant height discontinuity in the range imagery, this obstacle was not detected. The operator removed the rock for safety, but felt the vehicle could likely traverse over it. The run continued around the clump of trees and back up the hill to the starting point. Despite a great deal of slippage on this final leg of the journey, the vehicle ended up only thirteen feet away from the survey point where it started and only six feet away from where the planner thought it was.

For our final run, we used the complete goal list, and increased the maximum speed from 2 to 3 kilometers per hour. We also increased the radius for goal satisfaction from five meters to ten so that cumulative errors in position would have less significance. The run started out noticeably faster, and turns were more abrupt as well, appearing somewhat jerky. The vehicle proceeded without difficulty around the clump of trees, up the hill, around the Eiger, and then across the gully. Because of the larger goal radius, the vehicle crossed the gully sooner than it had on previous runs. Avoiding a few obstacles in the flat area, the speed dropped to 2.5 km per hour. The vehicle reached its goal in the field and then made a broad left-hand turn to get back across the gully. As in previous runs, the vehicle came back to the gully on the high side, but this time it saw the gully as an obstacle. The vehicle continued parallel to the gully, went past the desired crossing point but then got stuck in the mud while trying to turn right to reach its next goal.

## 5. CONCLUSIONS

We have demonstrated autonomous navigation of a robotic vehicle in natural terrain. Major performance improvements were accomplished between the August and December experiments. In the near future, implementation of the perception algorithms on the Warp systolic processor will hopefully provide an order of magnitude speedup in the processing time; and therefore, increase vehicle operating speeds and allow faster and more accurate detection of obstacles.

We learned a great deal about problems with planning, perception, and the vehicle itself from these experiments. We expect to be able to return with many of these problems resolved. Also, using new techniques and faster hardware we expect to increase the complexity of our scenarios and improve overall performance. In making these improvements, much attention will have to be devoted to compensating in software for inherent mobility problems of the ALV. Problems such as the limited clearance, inability to back up, and tendency to slip when turning will require special attention in both the planning and perception systems. Any correction of these problems in hardware would greatly enhance potential ALV operations. Meanwhile, perception development must focus on improved detection of gullies and detection of rocks surrounded by brush. For planning, local planning issues must receive greater attention to improve subgoal selection, and to help cope with unexpected situations. In general, we hope to work closely with Martin Marietta to try more complex scenarios, merging ALV road-following capabilities with our off-road capabilities, and demonstrating various mission-related behaviors.

## REFERENCES


M. Daily, J. Harris, D. Keirsey, K. Olin, D. Payton, K.Reiser, J. Rosenblatt, D. Tseng, and V. Wong, "Autonomous Cross-Country Navigation with the ALV," *Proceedings of DARPA Knowledge-Based Planning Workshop*, Austin, Texas, December 1987.

M.J. Daily, J.G. Harris and K. Reiser, "Detecting Obstacles in Range Imagery," *Proc. Image Understanding Workshop*, pp. 87-97, Feb. 1987.

M. J. Daily, John G. Harris, and Kurt Reiser, "An Operational Perception System for Cross-Country Navigation," DARPA Image Understanding Workshop, 1988.

J.S.B. Mitchell, D.W. Payton, and D.M. Keirsey, "Planning and Reasoning for Autonomous Vehicle Control," *International Journal for Intelligent Systems* 2(2), John Wiley & Sons,1987.

K.E. Olin, F.M. Vilnrotter, M.J. Daily, and K. Reiser, "Developments in Knowledge-Based Vision for Obstacle Detection and Avoidance," *Proc. Image Understanding Workshop*, pp. 78-83, 1987.

D.W. Payton, "An Architecture for Autonomous Vehicle Reflexive Control," *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA. 1986.

M.A. Turk, D.G. Morgenthaler, K.D. Gremban, and M. Marra, "Video Road-Following for the Autonomous Land Vehicle," *Proceedings of IEEE International Conference on Robotics and Automation*, Raleigh, North Carolina, 1987

R. Wallace, A. Stentz, C. Thorpe, H. Moravec, W. Whittaker, T. Kanade, "First Results in Robot Road-Following," *Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, California, pp. 1089-1095, August 18-23, 1985.

# Integration Effort in Knowledge-Based Vision Techniques for the Autonomous Land Vehicle Program [1]

Keith Price and Igor Pavlin

Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, CA 90089-0273

## ABSTRACT

In this paper we present a methodology and demonstrate some early results in the integration of knowledge-based image analysis programs. We specifically address the domain of complete three-dimensional motion analysis in the context of the Autonomous Land Vehicle. The integrated system exploits the strengths and minimizes the weaknesses of the individual techniques, resulting in performance which is considerably improved over the performance of any of the independently developed programs.

## 1 INTRODUCTION

Over the past several years the USC Computer Vision group has developed a number of component programs that can be applied to motion analysis in the Autonomous Land Vehicle (ALV). Thus, we have a number of separate programs (or collection of programs) developed by different people for different computer vision tasks [1], [2], [3], [S. Gazit these proceedings] with no strict requirements imposed on the developers as to what input, output and program parameters should be used. We will use the word *module* to refer to a collection of programs that are solving a particular task from the computer vision domain (for example, a collection of programs that find depth of environmental points using a pair of stereo images). Our current task is to construct a control structure that will use these different modules and enable them to cooperate in visual guidance of an ALV using general motion analyzing techniques.

We consider the integration to be an important effort for several reasons. Different feature extraction or matching techniques may work best in specific circumstances, thus a variety of modules for similar operations are necessary. Additionally, it is too costly and time-consuming to reprogram current modules into a coherent and unified computer vision program. Even if we would succeed in this effort, we would lose the generality of using the same basic modules for multiple applications. By using a variety of modules for similar operations (e.g., feature matching) we will develop techniques that can more easily accept other, newer, modules for the same or related processing steps. Therefore, we prefer using the current modules, at the expense of designing a control structure for them. In order to create the task configurations we must understand how the modules interact and the type of interfaces needed between modules.

The problem we face is not the problem of the top-down design which had divided the task into subtasks that will be later linked together. There, predetermined data structures enable easy module integration. We did not influence the design and the interaction of modules, although the modules may had been developed for similar domains end therefore could share similar input data.

There are several important issues we had to address in the integration effort. We chose not to create a completely general control structure and interface (such as a blackboard) because of the desire to quickly incorporate current results in component development. Here we present these issues in general terms and will later give some examples and initial results.

1. If we know how to combine different modules (in principle) what is it that we have to do (in practice) to make the combination of the modules work together? We call this an *interface* problem. The question is then to design interfaces in a manner that hides details of implementation of one module from another module without loosing the capabilities already present in either module.
2. How are we going to judge the *performance* of the combined modules once they are "sewn together?" If a human operator must assess the performance of the

[1] This research was supported, in part, by the Defense Advanced Research Projects Agency contract DACA76-85-C-0009 monitored by the U.S. Army Engineer Topographic Laboratories.

modules on a particular subtask, we will be not able to combine several modules to work automatically on a more complex task.

3. Can the system suggest (and eventually generate automatically) a *configuration* of the modules that will be the best for a given task? How can we incorporate the knowledge that people use when choosing a set of programs to perform some visual perception task?
4. Somewhat related to the previous two issues is: should we strive for a *static* or *dynamic integration*? In a static integration two modules are "hardwired" feeding input or output to each other (if a feedback is used) but essentially they are forced to work together independently of the input domain. Dynamic integration links modules at the run-time depending on the domain, module performances and the task in question. Dynamic integration is a much more complex problem and we are not in the position yet to attack this problem.

# 2 CONTROL STRUCTURE

In this section we outline the major design decisions that are made in the integration effort of different motion modules. In the next section we will present our results and the current state of the integrated software.

In the design of an integrated software for a particular task, the first step is to define module's input, output, its preconditions (range of parameters), purpose, efficiency, end expected quality of the results. The next step is the design of the control program, that will synchronize the work of motion modules. We have used a similar design strategy for our motion integration package as those found in design of software for automatic programming, in particular those found in work by Kant [4]. Although the task system is designed for the domain of motion analysis of the ALV, most of the decisions are equally applicable to a general purpose vision system.

## 2.1 Design Decisions

The system has four major components:

- The set of routines that specify modules and create a module configuration needed for particular task (task definitions).
- The set of routines that schedule and execute a particular task (task execution).
- The set of routines that create history of data and control flow. The user can then examine intermediate results, and rerun tasks perhaps using different modules or data.
- A user-friendly interface that allows the easy modification of input and output parameters and the easy design of new task configurations. It also helps in displaying the history of the run using images and data tables.

For each module we define the module components, function and required input and output data. We separate functionality of the module from its domain and range, so that we can create separate data and control flows [5]. This decision helps the user create a task configuration and build a control structure on top of the latter. Knowledge-based scheduling and execution require the separation of data and control flow for the same reason.

Task configurations are represented as graphs in which modules are nodes and paths are data and/or control flows. Each node (module) can have several input and/or output data ports because the type of the data required by modules greatly varies. In addition to the input and output data required for the module, there are also parameters for internal graphic displays and debugging information from the module. One of the important module characteristics is that they can be implemented in different programming languages (like C and Lisp in our case) and the data ports provide a convenient interface between these modules.

Modules can be configured using different control structures (loops, sequences, concurrent execution, conditional constructs, etc.). This means that we can use data feedback between modules, use several machines to run modules concurrently if needed, and make choices about module execution depending on the data they use. The design allows both forward-chaining and goal-directed reasoning which is needed in a more sophisticated task scheduling and execution environment.

The task history is a very helpful tool for rerunning the same set of module executions, examining the data (images, parameters) at each stage of execution, perhaps selecting a new set of parameters for a new run, and serving as a quick demonstration facility.

Task configuration, execution and history are accessible to the user through the graphic interface. A powerful graphic editor is used to help the user to: compose and decompose task configurations; enable task interruption or execution; examine, edit or save data used or produced at different stages of task execution; edit the global knowledge base; and enable task rerun. The graphics editor also has tools for data smoothing and data routing and conversion to and from different computational machines.

All the above program decisions are made in order to allow an interactive, user-friendly, problem-solving motion package that can be later used in a semi-automatic or automatic way to detect the environmental changes from images. Modularity of the design enables easy incremental addition or change of modules or data, and a greater flexibility and efficiency of the whole knowledge-based motion system.

## 2.2 Interface Problem

In this subsection we discuss issues in creating interfaces between different modules. We initially have developed a set of specific module interfaces rather than a single data transfer mechanism so that we can concentrate on computer vision rather than general system building. The other reason is that we currently have only a few modules for each subproblem and the design of interfaces between each pair of modules is not a costly design decision.

The long term effort requires that the interfaces between the two modules are general enough to handle not only the particular pair of modules, but a pair of classes of modules. A class of modules has elements that solve one specific problem of the vision, for example, all the modules that perform straight line extraction will be one class (say class A), and all the modules that find line correspondences will be in another class (say class B). The interface between any module from class A and any module from class B will be the same. The reason for such a design is that we would like to handle lines as semantic entities and not be concerned with detailed representation of the line.

The other important issue is a need to design these interfaces for possible use in a feedback loop. In these situations the output of the second module might be used to improve the performance of the modules that provided its input.

Interfaces hide details between the different requirements of different modules. In the example that we present in the next section, a motion estimation module requires the position of a region in several frames. On the other hand, a region matching module returns corresponding regions between two frames. The interface between matching and motion estimation modules accumulates the pairwise matches until enough are found for the motion estimation module. In other situations, the interfaces hide the details of data representation for different modules because we are only concerned with semantic notion of features and not its representation.

Sometimes an interface must account for missing data, and sometimes it should discard data that are not considered to be essential. We plan to equip the input and output data structures with procedures that will signal the absence of necessary data, so that the missing data could be recovered by calling some other module, or the user.

# 3 RESULTS

Our initial implementation provides a working prototype and a baseline system for testing of the integration framework. It is implemented in an object-oriented language (flavors). In this implementation, the initial control program that drives different modules is "hardwired," thus avoiding several important issues that usually appear in automatic programming. However, we still had to solve the interface problem. We have also implemented the most important parts of the user-interface.

As the initial step to integration, and to provide a convenient method to more easily test the motion estimation system, we have combined modules for feature extraction using the region segmentation techniques [6], feature matching using our region based matching system [2], three-dimensional motion estimation [3], and feedback of the image location prediction to the matching programs. These programs were written by different authors, without considering the need to integrate these specific programs into one system, thus some of the effort is required to transform the data produced by one system into data expected by the next. For example, the matching system provides a symbolic description of the two input images with links between them and the motion estimation program only requires a list of point correspondences for several frames. The list of points can be derived from the matching output.

This initial integrated system demonstrates the ability to combine different subsystems into one unified system. This prototype system has the following tasks (see Figure 1 for a description of the current system):

- **Image input:** Read the image sequence.
- **Image segmentation:** With large images and for time considerations, a subimage is segmented into regions by the histogram based segmentation program. These regions may, or may not, correspond to actual real-world objects, but are assumed to be single objects for the purpose of motion segmentation. All the images in the sequence are initially segmented. Features of individual regions and relations between regions are also computed. One of the features, the center of mass, is used later in the motion estimation module. This forms the symbolic description of the image.
- **Match the first image description to the second:** Initially, there is no information to guide the match, so the first few matching steps must use the general techniques with features such as intensity, size, shape, adjacencies, relative positions, etc. This produces a set of corresponding regions where a region in the first view is paired with a region in the second view.

- **Match the second image description to the third:** This step is the same as the previous one where general features must be used. At this point the translation estimation module used for generating predictions of future image plane locations for those regions that are tracked from image 1 to 2 to 3 (i.e., region X in image 1 is matched to region Y in image 2 which is then matched to region Z in image 3). The general motion estimation system requires one point in five consecutive frames, but three dimensional translation can be computed using only one point in three frames. Thus, the matching through three frames allows the prediction of the region location in the forth and future frames.
- **Continue the matching process for descriptions of image N to image N+1:** Since a motion estimate has been computed for some of the regions, the predicted position of the region in the next image can be used as a feature (the position) in the matching process. This allows greater motions to be easily handled by the later matches. The motion estimation programs (general estimation for 5 or more frames in a sequence and translation estimation for 3 or 4) are applied on each sequence of matching regions.

The motion estimation results are displayed in several forms at each stage, including the trajectory mapped back onto the image plane (using perspective projection), an orthographic projection of the trajectory viewed from the top, and another viewed from the side. These three displays are given in Figure 2, with the perspective view showing the motion of the four regions (grill (3,6), bumper (2,7), front shadow (1,4) and side shadow (4)) drawn for the six frames in the sequence and drawn on the next-to-last (fifth) frame. The computed motion projections for all the regions, except the side shadow, are shown for frames 1 through 5 (labeled 1, 2 and 3) and for frames 2 through 6 (labeled 4, 5, 6 and 7). The two orthographic views show that the motion is completely in the Z and X directions (see the side view motion where Y is almost constant for each region) and shows the trajectories of the regions in the correct relative positions. These three-dimensional trajectories are scaled to the dimensions of the focal plane of the camera since absolute scale can not be derived. The positions are also adjusted for the computed relative depth of the points as shown by the fact that the beginning locations for points 4-6 are closer to the camera (i.e., Z is smaller) than the beginning locations for points 1-3.

This version of the program was intended only as a test of the current component interfacing and to provide an outline for the future system.

# 4 FUTURE WORK

As we have seen in the previous section the initial motion integration package performs region segmentation, evaluates region correspondences not for a single pair but for many pairs of frames. Then an estimation motion module is called that determines the motion parameters of the ALV. Major areas for future work include using more feedback from motion estimation to matching and using feedback from both motion and matching to segmentation.

We also plan to add in the motion detection system another subsystem that uses a Hough-transform based module to detect preliminary line correspondences. The later will provide input to a module for more precise line-corresponden (these two might be connected via a feedback loop). The results of these two modules are to be fed into a third module that uses line-correspondences in several frames to detect motion of objects in the scene. The results on this subsystem will be reported later.

We will use the contour based matching approach [S. Gazit these proceedings] for direct input to the motion estimation programs and plan to combine it with the region based matching system. This will allow the detailed matching results using contours to be computed when the motion between frames is large.

These three examples demonstrate that we have different input situations in mind (some images suitable for region matching, some for straight, some for curved line matching), and that each group of modules will be used depending on the input data. That is an example of what is needed in a more general purpose vision guidance system.

# ACKNOWLEDGMENTS

We would like to thank Jasmina Pavlin for careful reading of the article and for useful discussions.

# References

[1] R. Nevatia and K.R. Babu. Linear feature extraction and description. *Computer Graphics and Image Processing*, 13:257–269, 1980.

[2] O.D. Faugeras and K. Price. Symbolic description of aerial images using stochastic labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(6):633–642, November 1981.

[3] H. Shariat and K. Price. Results of motion estimation with more than two frames. In *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, California, 1987.

[4] E. Kant. *Interactive Problem Solving with a Task Configuration and Control System.* Technical Report, Schlumberger-Doll Research, December 1987.

[5] D.R. Barstow. *Automatic Programming for Streams II: Transformational Implementation.* Technical Report, Schlumberger-Doll Research, August 1987.

[6] R. Ohlander, K. Price, and R. Reddy. Picture segmentation by a recursive region splitting method. *Computer Graphics and Image Processing*, 8:313–333, 1978.

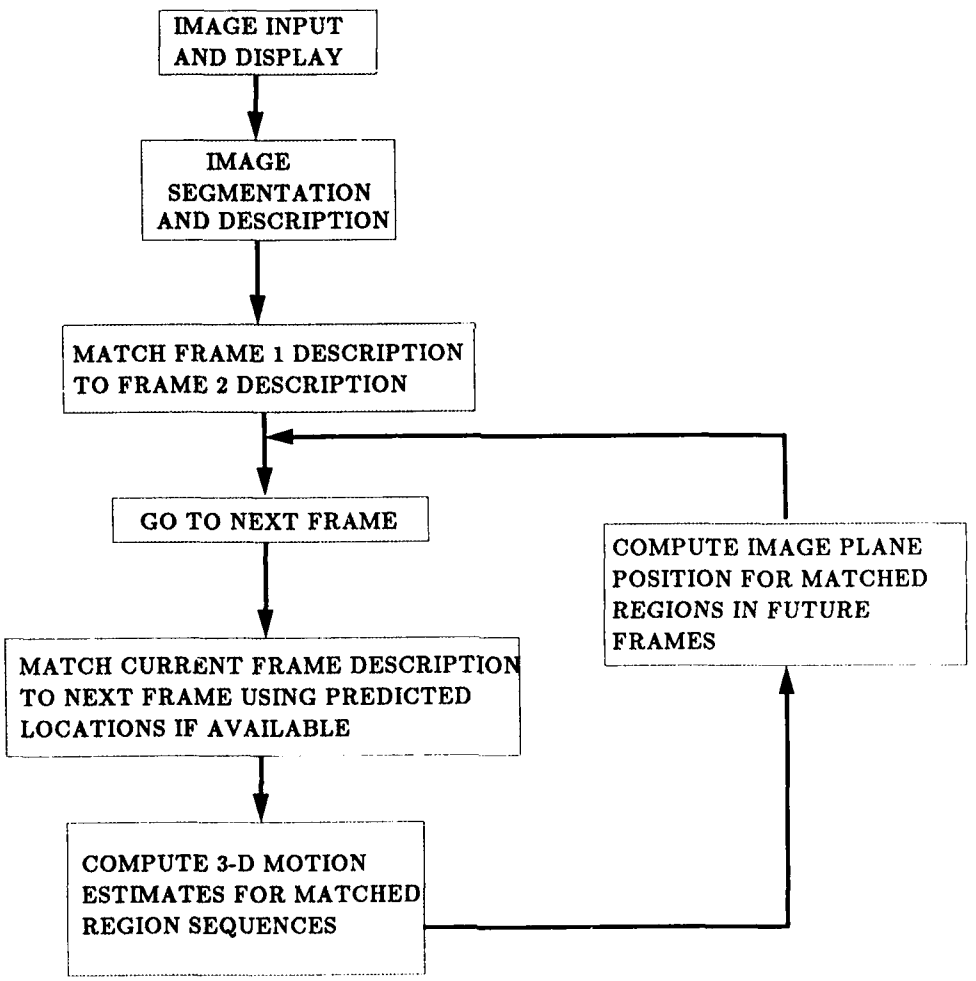
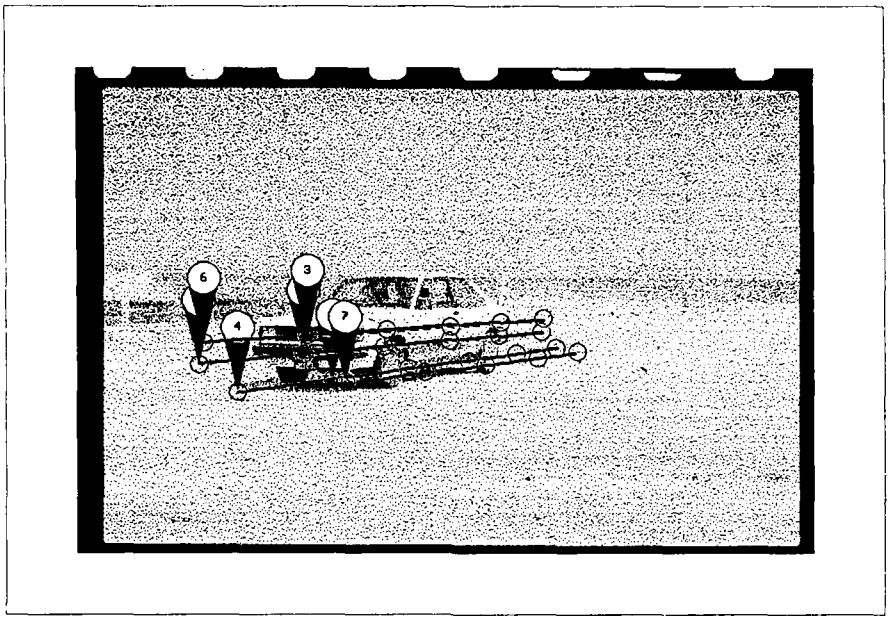
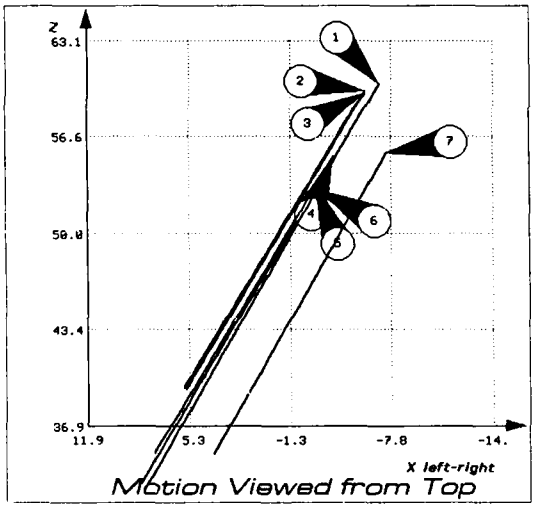



Figure 1: Description of Current System



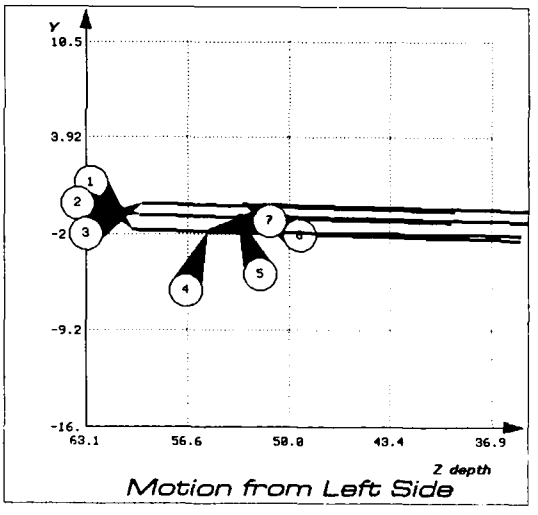




Figure 2: Results

# Contour Correspondences in dynamic imagery[1]

*S.L. Gazit and G. Medioni*

Institute for Robotics and Intelligent Systems
Depts of Electrical Engineering and Computer Science
PHE 204, MC0273
University of Southern California
Los Angeles, California 90089-0273

## Abstract

One of the fundamental problems in Computer Vision is the problem of establishing correspondences between two (or more) images. This paper addresses this problem and differs from previous approaches in the choice of matching primitives and criterion. We use *super-segments*, which are objects that can dually be represented either as groups of connected line segments or as chains of points, and we match *sections* of super-segments. The correspondence is based on similar shape and translation of matching sections. We use segment matching to limit search space, then apply a "chain matching" algorithm on sections. The method seems very robust, as demonstrated by the examples.

## 1 Introduction

Motion analysis is an important research area within the field of Computer Vision, and plays a central role in biological systems. Sophisticated mechanisms for observing, extracting and utilizing motion exist even in simple animals. Processing image sequence via computers has various applications in the medical, biological, industrial, military and other fields. Several approaches have been tried for computational analysis of motion from image sequences, and many of them need a set of matching points or matching features for the motion analysis. Therefore matching features between consecutive frames is an important step in motion analysis. This problem is more difficult than model or stereo matching which assume additional constraints such as shape preservance or epipolar lines, since objects may move, change shape, disappear, *etc.*

This paper is devoted to the problem of identifying corresponding points in two time varying images of a moving object (or objects). We assume that the maximal distance between corresponding features is known, to restrict the search space.

The major difficulty in matching arises due to the need for making *global correspondences*. A local point or area in one image may match equally well with a number of points or areas in the other image. These ambiguities in local matches can only be resolved by considering sets of local matches globally and imposing some preference criterion.

The various matching algorithms differ in the primitives used for matching, the method used for local matching and the method used for global matching, if any. The basic primitive in our algorithm is a *section* of a *super-segment*, where a super-segment is an object dually defined as both a connected list of edgel points and a connected list of line segments, and a section is some arbitrary portion of a super-segment. We use segment matching merely as an initial guide to section matching, so unlike other segment matching algorithms [15,16,1], we are able to use important features such as *continuity* along the super-segments and sections of arbitrary (not only linear) shape and length for matching.

For local matching we use shape similarity between sections of super-segments and for global matching we use relaxation in the translation space. We believe that using sections of super-segments removes many of the problems resulting from segment or edgel matching, since the *continuity* information can be better preserved than in segments, which have the collinearity constraint, or edgels which do not contain any continuity information and the area between sections is much more reliable than merely "similar orientation". Using sections of arbitrary shape yields, we believe, a better match than using only linear segments, since curvature implies a much stricter constraint on the match. We allow these sections to grow as long as the area between them remains small, so we get very long reliable matches, which correspond to object boundaries.

Section 2 describes previous methods, section 3 contains the description of the algorithm and section 4 presents our results and conclusions.

## 2 Existing Methods

### 2.1 Area Based Methods

Given two gray-level images, one would like to find a corresponding pixel for each pixel in each of the images, but the semantic information conveyed by a single pixel is too low

[1]This research was supported by DARPA contract DACA76-85-C-0009, order No. 3119 and monitored by the U.S. Army Engineer Topographic Laboratories

to resolve ambiguous matches, so it becomes necessary to consider an area or neighborhood around each pixel. Three types of schemes can be found:

Differencing Schemes ([12,11,13,20] and others), a simple and fast method which is widely used. These systems tend to fail if the motion is small, illumination is not constant or the moving object is not easily distinguishable, and can be confused by noise.

Correlation Schemes were applied to measure cloud motion [14], traffic control [23] and to radar images [13]. They tend to fail in featureless or repetitive texture environment, are confused by the presence of surface discontinuity in the correlation window, are sensitive to absolute intensity, contrast and illumination and their complexity heavily depends on the size of the correlation window.

Gradient Schemes [8,3,6] are widely used for calculation of optical flow, and assume that the motion between successive images is very small, so they are very sensitive to noise.

## 2.2 Feature Based Methods

These systems match features derived from the two images rather than the intensity arrays directly. The commonly used features have been edgels, linear line segments and corners (points of high curvature). These systems are usually faster than area based systems since they consider much fewer points, yet preserve significant points. On the other hand a lot of pre-processing is needed to extract the features, and due to the sparse data these systems do not produce a dense matched map.
Existing methods include graph matching techniques [10], relaxation [2,5], region matching [21] (useful when there is a significant change between frames, but tends to fail when there is occlusion) and more.

Matching edgels suffers from some of the limitations of the area based systems, since edgels are still very low-level. One isolated edgel is not very distinguished, so groups of edgels need to be taken in order to disambiguate matches.

A *line segment* (or just *segment*) is a linear approximation of connected edgel points and as such has some continuity information inherent to it, yet is local enough, so that the chance that a segment belongs to two physical objects is very small. Each segment contains information about its length, direction and position. Line segments are easy to represent and manipulate. Systems which match line segments exist mostly for Stereo image processing [16,1,17] and for image-model matching [15]. The main idea in these systems is essentially to locally slide two descriptions over each other for maximal fit. This approach guides our algorithm also.

# 3 Description of the Method

## 3.1 Primitives

We believe that the feature-based correspondense schemes have strong advantages over area-based schemes, because feature based systems consider much fewer points and are therefore faster than area based systems. By using features such as edgels, curves obtained by spatially linking these edgels, or even some approximation of these curves, the system is less susceptible to errors resulting from noise, change in illumination, *etc.* Curves formed of connected edgel points usually correspond to object boundaries, so the reduction in the amount of information does not necessarily mean reduction in the quality of the information.
Edgels however seem too local to be chosen as primitives. The advantages of *line segments* were discussed in section 2.2. We use segment matching as an initial step in our algorithm. When we try to evaluate matches however, the disadvantages of segments come into view: they are at best only approximations of the "actual" curve and sometimes a bad approximation (a circle for example). A curve may be

Sa - A super-segment
{a1, a2, a3, a4} - its segments.

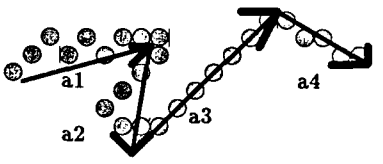



Figure 1: example of a super-segment

broken into segments differently depending on the segment fitting algorithm and on the amount of noise. The exact position of a match is not known because the segment matcher can only tells us that a line segment matches some other line segment but not which *pixels* actually match. Also most segment matching algorithms do not use the continuity between the segments.
For these reasons, we have decided to use curves or *super-segments* which are objects each having an ordered list of segments belonging to the super-segment and a description of its curve as a chain of the "actual" points of the super-segment. Also each segment knows which super-segment it belongs to and the position in the super-segment chain where it fits. An example of this "dual representation" is given in figure 1.

The input is obtained by computing zero crossings [9] of convolution with Laplacian of Gaussian masks [4] to get the edgels, then link the edgels and finally fit curves by piecewise linear segments [7]. The curves produced in this method are long, closed and relatively not noise sensitive, but their locations and shape may not be accurate (as explained in [4]).
Since continuity plays a very important role, we prefer zero crossings to other alternatives such as edgels produced by step masks [18] which have more accurate location, but the curves they produce are usually shorter and more noise

sensitive. Another reason for using zero crossings of LoG masks is the fact that in moving from a large mask to a smaller mask we get additional edgels but no edgels disappear, which may be useful for a top-down approach.

## 3.2 The matching algorithm

Follows is the general outline of our algorithm; the details are presented in the next subsections.

We match segments initially to obtain initial section segment matches, then divide each section point list into "pieces" (or sub-sections) and search for a best fit piece for each, trying to extend these pieces in the process. We evaluate these matches using relaxation in the translation space, and then remove overlapping (non-unique) matches, based on similarity in both shape and translation.

**The Matching Algorithm**

1. For each line segment in one image, find a subset of segments in the other image that can match this segment. (See section 3.3)
2. Match super-segments sections based on similarity in shape:
   (a) For every pair of maximal connected matching segment lists, define an initial match as the initial sections corresponding to these segment lists. (See section 3.4, step 1)
   (b) Divide each (left) section into smaller pieces, and find for each piece the "most similar" piece in any matching section. (See section 3.4, step 2)
   (c) Extend each match by adding adjacent points to the pieces matched, so that the similarity error measure is minimized. (See section 3.4, step 3)
3. Relaxation step: Remove matches for which not enough support exist, iterating until no matches are removed. (See section 3.5)
4. Remove overlapping matches. (See section 3.6)
5. Repeat once again steps 3, 2c and 4 (in that order).

In the next sections we discuss some of the steps in more detail.

### Notation

Let $\text{IMAGE}_1$ and $\text{IMAGE}_2$ be the images to be matched,
$A = \{a_i\}$ be the set of segments in $\text{IMAGE}_1$,
$B = \{b_j\}$ be the set of segments in $\text{IMAGE}_2$,
$S_A = \{s_{a_i}\}$ be the set of super-segments in $\text{IMAGE}_1$ and $S_B = \{s_{b_i}\}$ be the set of super-segments in $\text{IMAGE}_2$. We use this notation since $S_A$ ($S_B$) is actually a partition of $A$ ($B$).
Also let the *maximal disparity* $d$ be the maximal distance two corresponding features may have (measured in pixels).

## 3.3 Matching segments

The following algorithm computes for each segment $a_i \in A$ a subset of segments $b_j \in B$ that can match $a_i$:

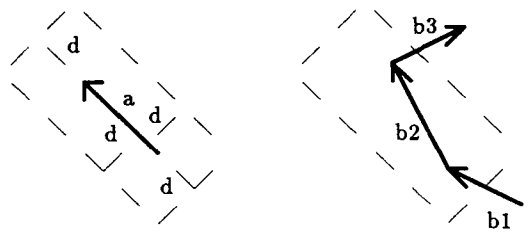
For each segment $a_i \in A$ define a window $w(a_i)$ in which corresponding segments from $B$ must lie, and define a similar window for segments in $B$. We have used a rectangular window parallel to the segment with width $2d$ and height $2d + l_i$, as $w(a_i)$. Note that $b_j \in w(a_i) \Rightarrow a_i \in w(b_j)$.

Let $a_i \in A, b_j \in B$ be two segments with orientations $\theta_i, \theta_j$ and length $l_i, l_j$ respectively. We say that $a_i$ matches $b_j$ if the following conditions hold:

$b_j \in w(a_i)$, $a_i$ and $b_j$ have "similar" orientation (the similarity measure is defined by equation 1), and the middle point of the shorter segment must intersect the window of the other segment.

$$|\theta_i - \theta_j| \leq \theta + \frac{\pi}{2} \cdot l \cdot (\frac{1}{l_i} + \frac{1}{l_j}) \qquad (1)$$

$\theta$ and $l$ are constants. We used $\theta = \frac{\pi}{6}$ and $l = 1$. (See [19]). Figure 2 contains an example of matching segments.




Left segment a matches right segments {b1,b2}, but not b3 (orientation).

Figure 2: example of matching segments

## 3.4 Matching super-segments based on shape similarity

**Note:** Depending on the context, a *super-segment* is an ordered list of segments or an ordered list of edgels comprising the segments.

**Definition 1** *The* position *of a point in a super-segment is the arc length of the point.*

**Definition 2** *A* section *of a super-segment is a connected list of edgels, which is a part of the super-segment (Note that segments and super-segments are a special case of section). A* piece *is a portion of a section.*

See figures 3 and 5 for an example.

The following algorithm computes initial section matching based on similarity in the shapes of the sections:

1. Initially two super-segments $s_{a_i}$ and $s_{b_j}$ can match if any of their segments match, and for each super-segment $s$ let $S_p(s)$ be the set of its possible matching sections, which are simply the maximal consecutive sub-chains which segments correspond. (see figure 3).

2. For every pair of matching sections $(P, Q)$, divide $P$ into pieces, so that $P = \{p_1, p_2 \ldots, p_k\}$. Using the segment matches, find a corresponding piece $q_i$ to every piece $p_i$ (in the same fashion as before). Note that the "actual match" for $p_i$ is probably contained in $q_i$, and therefore we need to search for it. We "slide" $p_i$ along $q_i$ searching for the match with the lowest similarity error measure.

   The similarity error measure is the *area of the match* over the total number of points in the two matching pieces squared. Figure 4 illustrates the idea. Appendix A contains a description of an efficient algorithm to compute the area between two matching sections.

3. For each match $(p_i, m_j)$ try to "extend" it by adding neighboring points as long as the error (computed in the same way as above) decreases. To reduce time complexity we used a binary search type extension (see figure 5).

## Notes

Matching each piece is done independently, so non unique matches are allowed, since we hope that at least one will "catch" its correct location. Dividing the initial large section into smaller pieces is necessary since the sections often do not fully match, but portions of them do (due to motion of objects, changes in illumination, occlusion or errors of the edge detector). Extending the matches is necessary, as long matches are much more reliable than shorter ones, so good matches are better distinguishable from bad ones.

We chose a bottom up approach, in which we break the initial matching sections into small enough pieces and try to match each such section, then try to extend the match as long as the shape of the curve is similar enough (Another option is to determine where is the best place to "break" a super-segment, but this is complicated, since it requires finding corners, junctions and other high level features, and may fail when we have occlusion and motion). The size of the initial pieces was chosen as $\frac{l}{\log(l)}$, where $l$ is the length of the shorter of the two initially matching sections. This was a compromise figure between having a constant number of pieces per section (which penalized long sections) and having a constant piece size (which penalized short sections).

## 3.5 Relaxation step

In this step we evaluate matches using a global criterion, namely similarity in 2-D translation of neighboring matches (similar to other matching algorithms). We discard a match if the total number of points in matches which support it is below some threshold value (defined later). The support is based on "similar" average translation within some neighborhood.

**Definition 3** *$s_{a_k}$ is a* neighbor *of $s_{a_i}$ if the distance between their closest points is less than the maximal disparity.*

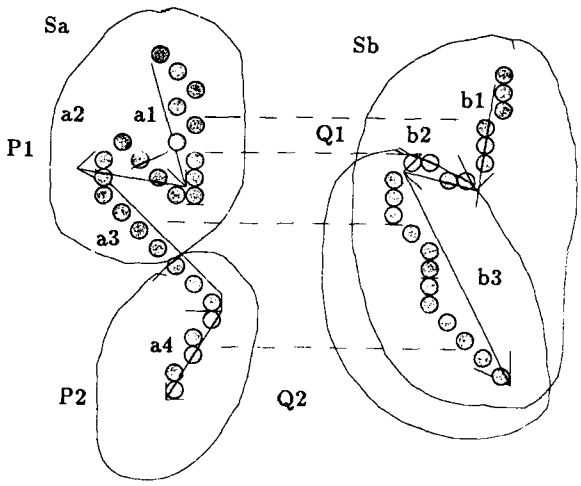
The neighbors can be computed in the same way as the initial matches were computed in the previous section.

Let $d_e$ correspond to the expected error in the "real" position (after compensating for the motion) of the object ($d_e$ should be 0 if no rotation, expansion or errors of the edge detector occur, but is usually larger). We used $\sigma$ (the space constant of the LoG filter) when we did not expect a major change in the shape (due to expansion), since the error in the position of the edgels depends on $\sigma$. Otherwise we used the maximal disparity (supplied by the user).

Let $M_{ij} = (p_i, m_j)$ be some match with with translation $(\overline{x_{ij}}, \overline{y_{ij}})$ where $p_i$ is a section of a super-segment $s_{a_i}$ and $m_j$ is a section of a super-segment $s_{b_j}$. A match $M_{hk} = (p_h, m_k)$ can *support* $M_{ij}$ if $|\overline{x_{ij}} - \overline{x_{hk}}| \leq d_e$ and $|\overline{y_{ij}} - \overline{y_{hk}}| \leq d_e$, it is not too short (its length is at least $\sigma$), either $s_{a_k}$ is a neighbor of $s_{a_i}$ or $s_{a_i}$ has no neighbors and either $s_{b_k}$ is a neighbor of $s_{b_j}$ or $s_{b_j}$ has no neighbors. Note that $M_{ij}$ can support itself.

$M_{ij}$ is kept if the total length of the matches that can support it is above a certain threshold or one of these matches is long enough. (our threshold was half the sum of the average length of the matches and the length of the longest match, and a match was long enough to support alone if its length was at least $2\sigma$.)

We iterate until no matches are removed.

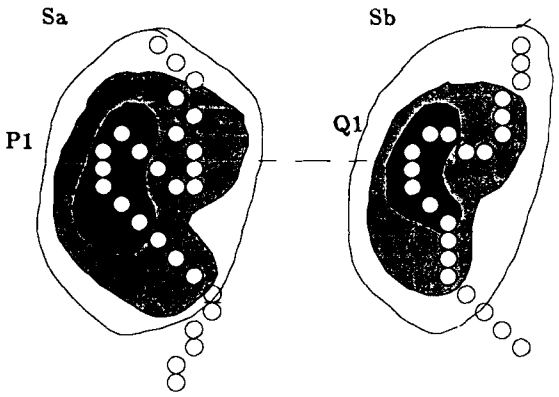



Sa, Sb - super-segments
{a1,a2,a3,a4} - segments of Sa, {b1,b2,b3} - segments of Sb.
P1, P2 - sections of Sa, Q1, Q2 - their initial matching sections.
Matching is based on segment matching : a1-b1, a2,-b2, a3-b3, a4-b3.

Figure 3: example of matching super-segment sections


Score of the match (P, Q) is the area between them (colored) over the total number of points in the two sections squared.

Figure 4: Area between two sections $(P, Q)$ ($Q$ translated to start where $P$ starts)




Extension of the match:
Initially the inner sections only match, then they are extended until score becomes worse.

Figure 5: example of matching super-segment sections

## 3.6 Removal of overlapping matches

Let $M_1 = (P, Q)$ and $M_2 = (O, R)$ be two matches. We say that $M_1$ and $M_2$ overlap if either $P$ and $O$ are sections of the same left super-segment $s_{a_i}$, and have points in common, or $Q$ and $R$ are sections of the same right super-segment $s_{b_j}$, and have points in common.

Assume (w.l.o.g.) the first case, then two possiblities exist:

- Partial overlap
- Complete overlap.

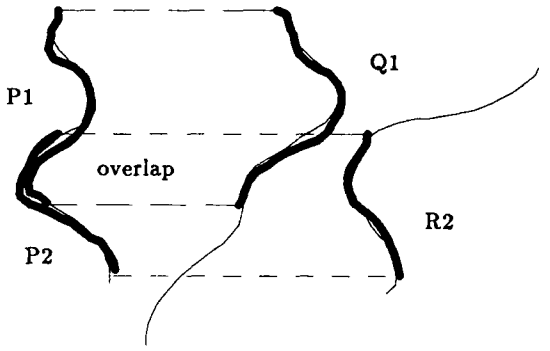
The two cases are illustrated in figure 6. In both cases the solution is to take the better match and the remainder of the other match. In the example of figure 6, we take matches $M_2$ and the remainder (the non overlapping portion) of $M_1$. In the second case, we prefer $M_3$.

To evaluate matches we try to use both the similarity in shape and in translation. We say that a match $M$ is better than a match $N$ if the score of $M$ (as computed by the previous step) + (1000 over the number of points in supporting matches) is lower than that of $N$.

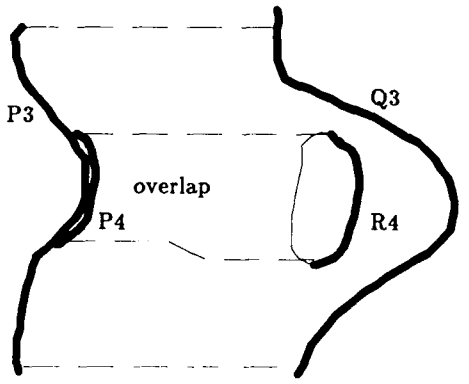
## 3.7 Why repeat the previous steps ?

In the algorithm to match sections based on shape similarity, each section was matched and extended independently. Therefore we expect a lot of overlapping matches. For example, consider the case of matching two identical super-segments of length $l$: we divide the first to $\log(l)$ sections and then extend each independently, so we end up with $\log(l)$ identical matches. The overlap-removal algorithm will remove $\log(l) - 1$ of these matches. In all our experiments the number of matches was significantly reduced after this step. If many matches were removed, there can now be matches with not enough support (see section 3.5), so we need to apply the relaxation again. If two overlapping matches were divided by the overlap-removal algorithm, and then one of them was removed by the relaxation, the remaining one can now be extended again. Therefore we apply the step to extend matches again and then remove the new overlaps created by extending the matches.

Theoretically we can then repeat the relaxation again, then extending and so on. However, the changes at this stage are expected to be minor, and since we cannot guarantee convergence, we do it only once.




M1 = (P1,Q1) partially overlaps M2 = (P2,R2)




The match M3 = (P3,Q3) contains the m :tch M4 = (P4,R4).

Figure 6: The two overlap possibilities

# 4 Results

We applied our algorithm on a number of real images, indoor as well as outdoor scenes. As long as the shapes of objects in the scene (as projected in the image) did not change much, the results were very good. The results are shown by displaying only those points for which a match was found, and drawing an arrow to the closest point in the other section (after translating to start at same location). The arrow is drawn for every fifth point in a matching section for each matching section), for clarity.

We give five examples, in each of which
figures (a),(b) contain the original images,
figures (c),(d) contain the super-segments obtained from the zero crossings of the convoived images
and figure (e) contains the result of the matching.

1. Figures 7 contain two $512 \times 512$ pixels images taken from a sequence of a road scene ($\sigma = d = 10$). Both the observer and the other car are moving.
2. Figures 8 contain two $512 \times 512$ pixels images of a car crossing the observer view-point ($\sigma = 10, d = 30$). The algorithm performed well on the image, even though the disparity was large, which shows that the location of the match does not matter much, as long as the shape does not change significantly between the frames.
3. Figures 9 contain two $256 \times 256$ images of an office scene ($\sigma = 5, d = 10$). The camera faces the direction of motion, so we expect objects to expand.
4. Figures 10 contain two $256 \times 256$ images of a corridor ($\sigma = 5, d = 10$). The camera faces the direction of motion, so we expect objects to expand.
5. Figures 11 contain two $256 \times 256$ images of an outdoor scene of trees ($\sigma = 5, d = 10$). This is a lateral motion case, which is made hard by the large disparity differences, and therefore most stereo algorithms will not match it successfully. We did not use the knowledge that motion is only lateral and allowed search in all directions, yet the algorithm was able to match the scene quite well. Using the epipolar constraint would probably improve the result.

The images in Figures 9, 10 and 11 were obtained from SRI International, courtesy of Dr. Bolles.

The program to implement our algorithm was written in *Common LISP* on a *Symbolics* Lisp Machine.

# 5 Conclusions and Future Work

We have shown an algorithm to compute correspondence between 2 frames with very few constraints. We suggested the use of super-segments and sections of super-segments. Correspondence was based on shape similarity between matching sections and on translation similarity between matches, and demonstrated some results on a number of real images.

The advantages of our method were discussed in the previous sections: the use of continuity and sections of arbitrary shape and size in matching, the use of length of a match, evaluation of matches based both on shape and on common translation.

Some comments are in order:

- The algorithm has a very heuristic flavor.
- The algorithm performs best on long curved contours, so it seems to best fit for matching zero crossings curves or region contours. We plan to try applying it to regions and to curves of the same image, processed with *different* LoG masks.
- We may get better results for stereo pairs by applying this algorithm with the epipolar constraint, as it can handle sharp changes in disparity, as demonstrated by example 11.
- The computation is made in 2-D only, but we can find the actually corresponding points using areas of high curvature or even the simple method we used for displaying the results (a left point matches the closest point in the translated matching right section). These point-to-point matches can be used for motion estimation in 3-D. We are currently working on using the Motion Estimation algorithm developed in [22]. This algorithm uses matching points in three or more frames to estimate 3-D motion and location of points in frames as well as give some error measure to the match. Since using the Motion Estimation algorithm requires matches in multiple frames, an algorithm to combine the results of matching pairs of images will be useful.

# Appendix: Computing area between two sections

The following is a general idea of the computation of area between two matching sections (some of the details have been left out). The idea is to translate the right section to have same starting point as the left section, and add points to ensure that the last point is also the same (we might have to remove remaining points from the sections). We get two sections which start and end points are the same, so we have a *cycle*. We find all simple cycles (cycles which do not cross themselves) and compute the area for each by a simple procedure. Figure 4 illustrates the idea.

The algorithm:

Assume sections $P = (P_1, P_2, \ldots, P_k)$ and $Q = (Q_1, Q_2, \ldots, Q_l)$ are possible matches, where $P_i = (x_i^p, y_i^p)$ and $Q_i = (x_i^q, y_i^q)$. Translate $Q$ to start at $P_1$. Define an *intersection point* as a point $P_i$ such that there is a point $Q_j$ in the translated $Q$, such that $P_i = Q_j$ (Note that $P_1 = Q_1$). Find all intersection points (this can be done linearly by drawing the left section in the plane and traversing the translated right section). The points of the two sections which lie between two adjacant intersection points form simple cycles. The sum of the areas of the simple cycles is the area of

the match. We compute it by assigning every cycle point $(x, y)$ a score $s = x_n - x_p$ where $x_n$ and $x_p$ are the $x$ coordinate of the next and previous points on the cycle. Let $A[x] = ((y_1, s_1), \ldots, (y_r, s_r))$ a sorted scan line. The area of the cycle along the scan line is the sum of all distances for which the accumulated score is non-zero.

The algorithm is linear in the number of points of the two sections, except where we sort the rows. This step requires $r \log(r)$ time, where $r$ is the number of points of this scan line. It will usually be a constant though, since points along horizontal lines have score zero and therefore do not affect the sum and can be eliminated.

## References


[1] N. Ayache and B. Faverjon. A fast stereovision matcher based on prediction and recursive verification of hypothesis. In *Proceedings of the 3rd IEEE Workshop on Computer Vision: Representation and Control*, pages 27–37, Bellaire, Michigan, Oct. 1985.

[2] S. T. Barnard and B. Thompson. Disparity analysis of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):333–340, July 1980.

[3] C. Cafforio and F. Rocca. Methods for measuring small displacements of television images. *IEEE Transactions on Information Theory*, 22(5):573–579, Sept. 1976.

[4] J. S. Chen and G. Medioni. Detection, localization and estimation of edges. In *Proceedings of Workshop on Computer Vision*, IEEE, Miami Beach, Florida, Nov. 1987.

[5] L. Dreschler and H.-H. Nagel. Volumetric model and 3-d trajectory of a moving car derived from monocular tv-frame sequence of a street scene. In *International Joint Conference on Artificial Intelligence*, Vancouver, Canada, Aug. 1981.

[6] C. L. Fennema and W. B. Thompson. Velocity determination in scenes containing several moving objects. *Computer Graphics and Image Processing*, 9:301–315, Apr. 1979.

[7] S. Gazit and G. Medioni. Accurate detection and linking of zero crossings. *Submitted to CVPR 88.*

[8] B.K.P. Horn and B.G. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–204, 1981.

[9] A. Huertas and G. Medioni. Detection of intensity changes with subpixel accuracy using laplacian-gaussian masks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(5):651–664, Sept. 1986.

[10] C. J. Jacobus, R. T. Chien, and J. M. Selander. Motion detection and analysis by matching graphs of intermediate level primitives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(6):495 510, Nov. 1980.

[11] R. Jain, W. N. Martin, and J. K. Aggarwal. Segmentation through the detection of change due to motion. *Computer Graphics and Image Processing*, 11(1):13–34, Sep. 1979.

[12] R. Jain, D. Militzer, and H.-H Nagel. Separating nonstationary from stationary scene components in a sequence of real world tv-images. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 612–618, Cambridge, Mass, Aug. 1977.

[13] R. Jain and H.-H Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):204–214, Apr. 1979.

[14] J. A. Leese, C. S. Novak, and V. R. Taylor. The detection of cloud pattern motions from geosynchronous satellite image data. *Pattern Recognition*, 2:279–292, Dec. 1970.

[15] G. Medioni and R. Nevatia. Matching images using linear features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):675–685, Nov. 1984.

[16] G. Medioni and R. Nevatia. Segment-based stereo matching. *Computer Vision, Graphics, and Image Processing*, 31:2–18, 1985.

[17] R. Mohan, G. Medioni, and R. Nevatia. A fast stereovision matcher based on prediction and recursive verification of hypothesis. In *In Proceedings of the 1st ICCV*, IEEE, 1987.

[18] R. Nevatia and K.R. Babu. Linear feature extraction and description. *Computer Graphics and Image Processing*, 13:257–269, 1980.

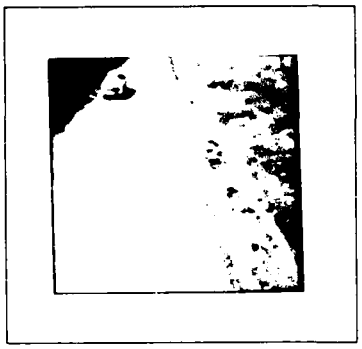
[19] R. Nevatia and K. Price et al. *Research in Knowledge-Based Vision Techniques for the ALV Program.* Technical Report 201, IRIS, University of Southern California, Los Angeles, California, Sept. 1986.

[20] R. M. Onode, N. Hammano, and K. Ohda. Computer analysis of traffic flow observed by substractive television. *Computer Graphics and Image Processing*, 377–399, Sept. 1973.

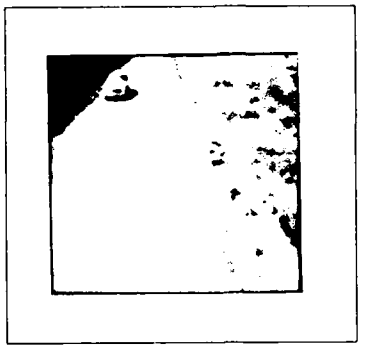
[21] K. Price and R. Reddy. Matching segments of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(1):110–116, Jan. 1979.

[22] H. Shariat. *The Motion Problem - How to use more than two frames.* PhD thesis, IRIS, University of Southern California, Los Angeles, California, Oct. 1986.

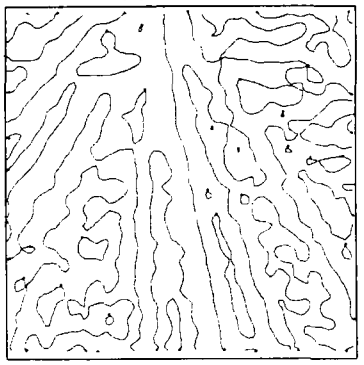
[23] K. Wolferts. Special problems in interactive image processing for traffic analysis. In *Proceedings of the 2nd International Joint Conference on Pattern Recognition*, 1974.



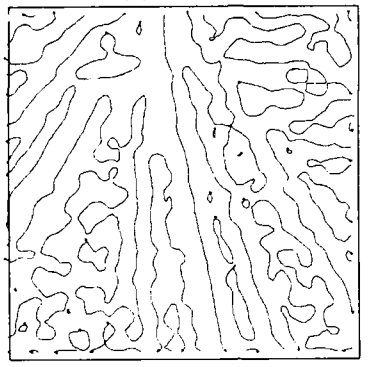
(a) Frame 1



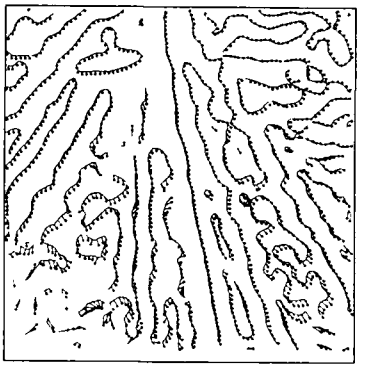
(b) Frame 2



(c) Zero crossings of (a)

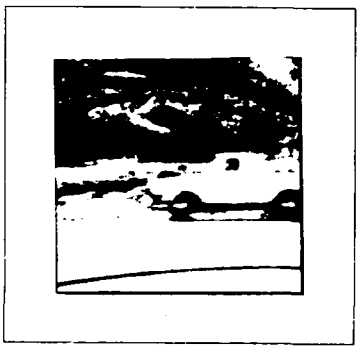


(d) Zero crossings of (b)

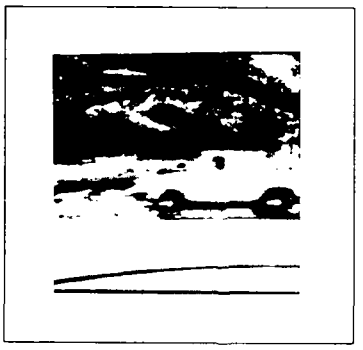


(e) Matches of (c),(d)

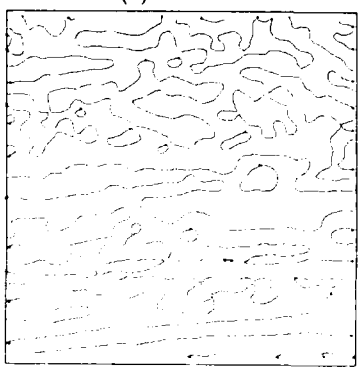
Figure 7: Advancing car



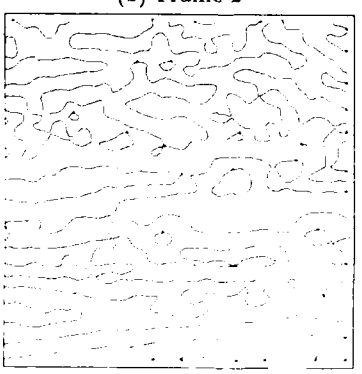
(a) Frame 1



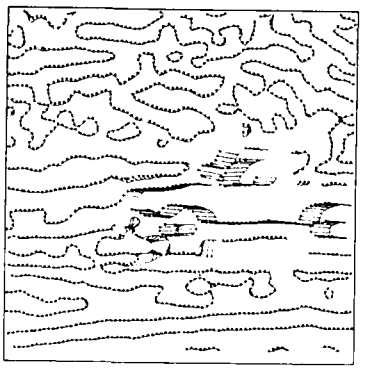
(b) Frame 2



(c) Zero crossings of (a)

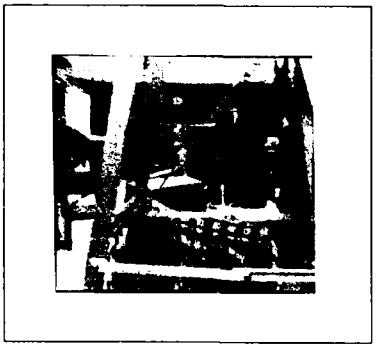


(d) Zero crossings of (b)

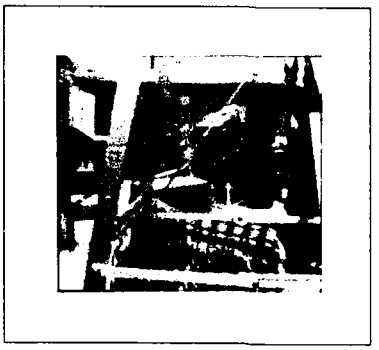


(e) Matches of (c),(d)

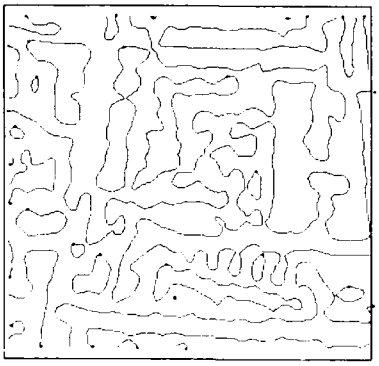
Figure 8: Crossing car



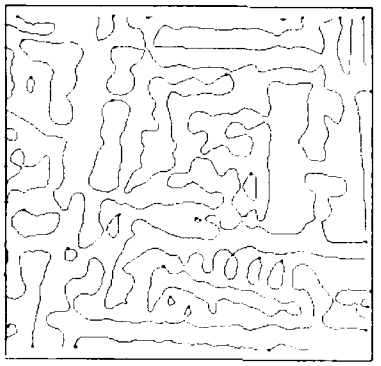
(a) Frame 1



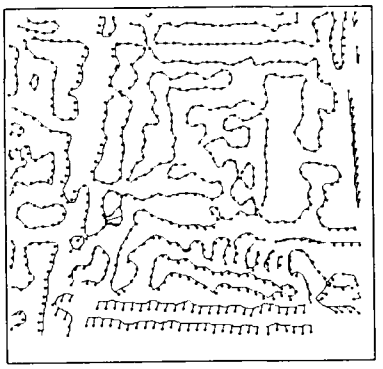
(b) Frame 2



(c) Zero crossings of (a)

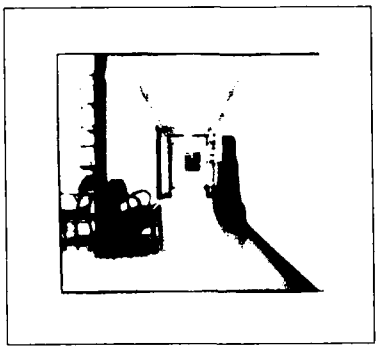


(d) Zero crossings of (b)

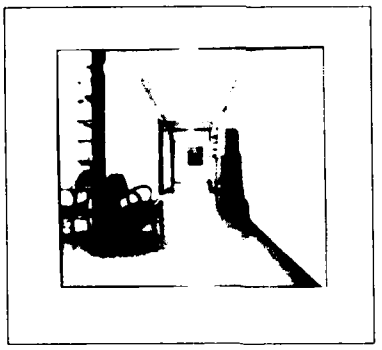


(e) Matches of (c),(d)

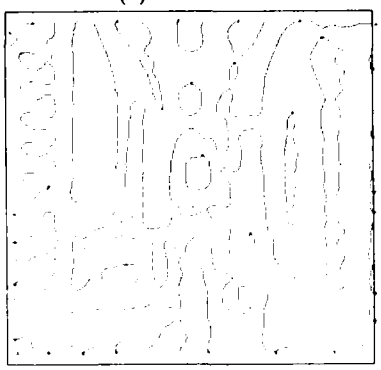
Figure 9: Office Scene



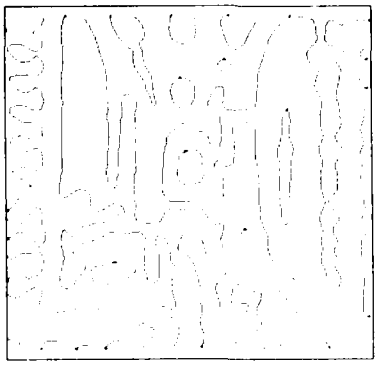
(a) Frame 1



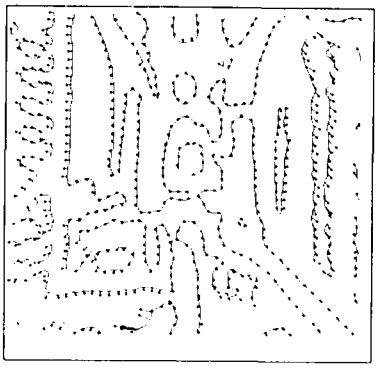
(b) Frame 2



(c) Zero crossings of (a)



(d) Zero crossings of (b)

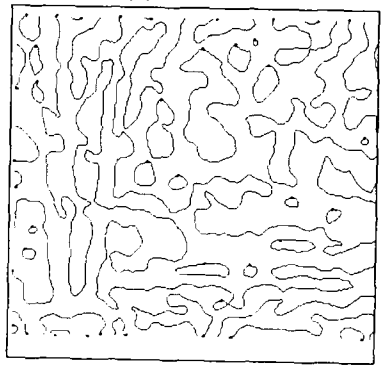


(e) Matches of (c),(d)

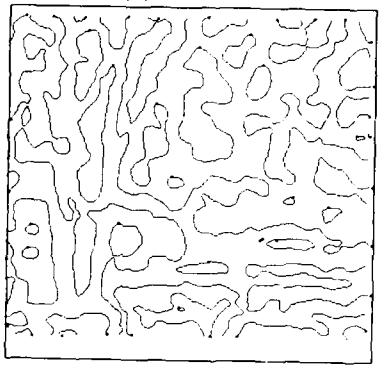
Figure 10: Hallway

(a) Frame 1

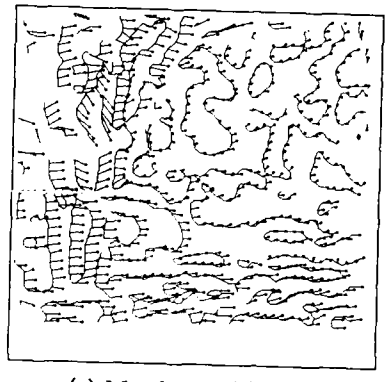
(b) Frame 2



(c) Zero crossings of (a)



(d) Zero crossings of (b)



(e) Matches of (c),(d)

Figure 11: Outdoor Scene (trees)

# Spatio-temporal Analysis of an Image Sequence with Occlusion[1]

*Shou-Ling Peng and Gérard Medioni*

Institute for Robotics and Intelligent Systems
School of Engineering
University of Southern California
Los Angeles, California 90089-0273

**Abstract**

We present a method to analyze a sequence of image frames taken close apart in time, forming a spatio-temporal volume. We first detect edgels in each spatial image, then analyze individual *slice* centered on each edgel and taken in the temporal direction. The combination of results from these multiple slices enables us to compute an estimate of the velocity in the direction normal to the tagent to the curve. This resulting normal velocity field may then be used to compute the real velocity field and to perform object segmentation. In contrast to most previous approches, the system is quite robust, it is capable of handling occlusion as well as disocclusion as they are explicitely modelled. We present results on synthetic data consisting of two objects moving with occlusion and results on real image sequences.

## 1 Introduction

Motion understanding is one of the most important visual functions, and has numerous applications in robotics and industrial automation. The information extracted from this process include segmentation, range, velocity, and so on. Motion therefore plays a basic role in the understanding process. It seems very reasonable that animals have perceptual systems or subsystems purely based on motion [25]. Some animals are known to shake their heads to gather information for hunting. Visual processing is neither a pure bottom up processing nor a pure top down one. Communication and feedback are necessary between high level and low level processing. In this paper, we try to identify and simulate a low level, local mechanism for motion detection.

[1]This research was supported by the Defense Advanced Research Projects Agency and was monitored by the U. S. Army Engineer Topographic Laboratories under contract DACA76-85-C-0009

The method to perform motion detection and understanding introduced in this paper is basically domain independent. It is able to calculate flow from discrete images and to separate objects based on their motion alone. The procedure developed here is not computationally intensive, and the information needed is only local in nature, making a VLSI implementation possible [15].

The following assumptions and restrictions are made regarding the observed sequence of images [20] :

1. *Maximum velocity*
   the operator is only sensitive to a finite range of velocity. An object can move at most $V \cdot dt$ between two images taken $dt$ time units apart,
2. *Small velocity change*
   it is a consequence of physical laws and the assumption of high sampling rate,
3. *Small shape change*
   each object is either rigid or is changing its shape slowly,
4. *Common motion*
   objects are spatially coherent and therefore appear in images as regions of points sharing a *common motion*,
5. *Causality*
   objects cannot appear or disappear suddenly.

The principle behind our approach is to find the velocity components of an edge point along several different directions and estimate its *normal* velocity, that is the velocity in the direction normal to the direction of the edge, subject to the constraints listed above.

The next section is a brief review of previous work in motion analysis, in which three different approaches are discussed and compared. In section 3, the basic idea of combining spatial and temporal information is introduced. The method we are proposing to segment objects from a sequence of images is discussed and formalized in the section as well. Several results are given in section

4 to illustrate how the method works. There are results on both synthetic and real image sequences. Finally, a summary of remarks is contained in section 5.

# 2 Previous Work

Motion analysis is a strong research area in computer vision. The key to understanding of image sequences lies in the analysis of differences and similarities between consecutive time frames. The approaches taken differ in the type of primitives used for matching, the criteria used to resolve ambiguities and the number of frames in the sequences. There can be broadly classified as follows:

## 2.1 Feature-based Approaches

This approach is probably the most intuitive if identifiable spatial features can be extracted and then the correspondences are possible to establish. A variety of possible features have been tried: points, line segments [17], blobs, local edges [12], vertices [2], local maxima of variability [3,18], local statistics [25], extrema of the local grey value curvature [7], corners[6,24], regions [21,27] or even recognized objects. Good features are those which can minimize the effect of illumination and geometric changes. The higher the level of descriptions at which matching is attempted, the less ambiguous the matching process will be, but this gain may be offset by the errors and deficiencies of the current programs producing those descriptions. The sampling rate may be large as long as the features are still in present the images. The accuracy is high if a sharp and localized feature is tracked, but such desired feature may be hard to find.

The extracted features of images are then matched to calculate a set of disparity vectors for the sequence. The correspondence is established based on a metric affinity function as well as a group mapping criterion. The best match is found based on an optimization criterion. Criterion functions can range from simple cross-correlation [9] to sophisticated graph-matching procedures [12]. The matching process is computational expensive. Methods such as coarse-to-fine resolution matching [10] may be used to speed up the process.

## 2.2 Intensity-Based Approaches

This approach can be subdivided further in three:
The first type is a *differencing scheme* which is done by subtracting one image from the other and thresholding the result. The clusters of points in the difference image correspond to moving objects. By ignoring the stationary background, the computational resources are focused on the moving objects [14]. This scheme prefers large motion so that the interesting objects are far enough not to overlap in position in different images, because the interior of homogeneous regions do not generate a difference. It fails when the observer is moving or when the illumination is not constant.

The second type is a *correlation scheme*. A patch of the image is used as a template and cross-correlated with other images. The peak value indicates a match in intensity and defines a disparity for the image patch [13]. This scheme suffers from the following limitations [17]:

1. it requires the presence of a detectable texture within each correlation window, and therefore tends to fail in featureless or repetitive texture environment.
2. it tends to be confused by the presence of a surface discontinuity in a correlation window.
3. it is sensitive to absolute intensity, contrast, and illumination.
4. it gets confused in rapidly changing depth fields (e.g., vegetation).

The third type is a *gradient scheme* which is widely used for the calculation of optical flow [8]. If $I(x,y)$ denotes the intensity function of the image, then the following holds:

$$-\frac{\partial I}{\partial t} = G_x \cdot u + G_y \cdot v \qquad (1)$$

where $\frac{\partial I}{\partial t}$ is the temporal intensity change at position (x,y); $G_x$ and $G_y$ represent the intensity gradient at the image point; and $u$, $v$ are local velocities in the x and y directions, respectively. Since $\frac{\partial I}{\partial t}$, $G_x$ and $G_y$ are all measurable by the observer, $u$ and $v$ can be determined by the above relation.

Anandan suggested a framework to compute dense field of displacement vectors with associated confidence measures [1]. In general, intensity-based approaches are faster at the cost of high data volume. The images must be analyzed for every few pixels of displacement, which means a high sampling rate. This approach allows complex shape changes and introduces the many-to-one match problem. It is also very noise sensitive and less accurate due to ambiguity of local measurements. A VLSI analog circuit was designed at Caltech to implement equation (1) [26]. The local ambiguity due to the *aperture problem* is handled by a constraint-solving circuit.

An interesting experiment demonstrates that the feature-based approach is a high level processing while the other one is low level [22], see fig. 1. A solid square is shown in the center against a dark background and is then replaced with an outline square on the left and a

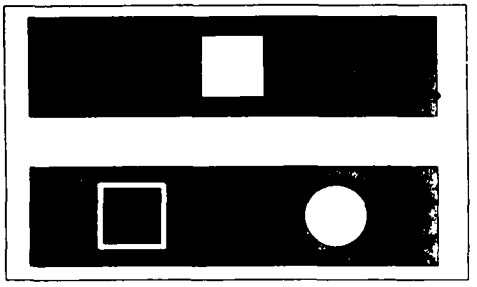


Figure 1: Features of Objects

solid circle on the right.

The viewer who is confronted with these images usually sees the square moving toward the circle rather than toward the outlined square, but when the images are presented slowly and there is time to scrutinize the image, then the perception is that the square moves to the outline square. This suggests that regions of low spatial frequencies (smooth intensity change) are more likely to be detected initially, which would suggest that intensity processing is performed by a preprocessor.

### 2.3 Image-Sequence-Based Approach

There is still another approach using a sequences of closely spaced images. This approach has received little attention until recently because of the huge amount of storage and computation involved. A solid of data called *spatio-temporal data*, with time as the third dimension, was introduced by Bolles and Baker [4]. It is constructed by a sequence of images close enough that none of the objects moves more than a pixel or so between frames. The *epipolar-plane image*, or EPI, is a slice taken from the spatio-temporal data along the temporal dimension. They used EPI to simplify the matching phase in stereo analysis. Consider a simple lateral motion in which a camera moves from right to left along a straight track and takes pictures at constant distance with its optical axis orthogonal to its direction of motion. Any feature point P describes a linear trajectory on the EPI because the only motion is horizontal and constant. The slope of the line determines the distance from the point to the camera. Occlusion is also immediately apparent in this representation. Those linear trajectories are then extracted by a non-directional Laplacian-Gaussian filter which treats the time domain the same as the horizontally spatial domain. Therefore the edge features are mixed with the intensity discontinuities due to occlusions. An extended work using projective duality is proposed in [16], which is expected to generalize the linear camera motion to an arbitrary one. This will still be applicable only if the camera path is known, and if the scene is frozen. To generalize this idea for motion analysis, consider the case where the camera is fixed. The motion of an image point still gives a continuous trajectory in the spatio-temporal data but it is not necessarily to be a straight line and, in general, does not fall on any EPI. A new approach is to be discussed in the next section to recover the trajectory called a *path* in order to derive the motion information. In contrast to some previous methods which require the acquisition of the complete spatio-temporal volume before processing is done, the method described here provides estimation after a few frames, and refining them as more frames come in. It therfore makes better use of storage and processing is faster.

## 3 Description of the Approach

From many biological experimental evidences, the primitive animal visual processing can be modeled as a nonlinear system which is a function of time and space. The system function is basically a composition of a spatial bandpass filter and a temporal bandpass filter. The central frequency and bandwidth define the range and sensitivity of its motion detection ability. The filtering effect permits to find the highest correlations in both temporal and spatial domain.

The goal of this paper is trying to devise a primitive parallel process which is able to extract motion information locally from the intensity image. The extracted information is passed to the higher level for a globally consistent interpretation.

### 3.1 Basic Idea

In many low level biological visual systems, edges are always one of the most useful features detected by the front-end preprocessing. When we look at a scene with moving objects, we are first alerted by the moving edges and then the movements propagate into the interior of the corresponding regions. At this moment, our internal representation of the scene becomes a bunch of surface patches associated with velocities. Those surfaces may be matched with our internal models to recognize moving objects. Motion is not the only cue human uses to visualize the world, but some other life forms do rely on motion exclusively, e.g. the predacious activity of the frog. They prey only on moving worms or insects and their attention is never attracted by stationary objects.

The motion information we want to extract is the normal flow associate with the edge elements. The aperture effect restricts us so that only one component of the motion in the 2-D image can be estimated.

Assuming a dense image sequence is available, the method chosen for the normal flow estimation is basically a spatio-temporal analysis on the *slices* constructed form the image sequence. A *slice* is a collection of **L** 1-D images of width **2W** taken from **L** successive frames in

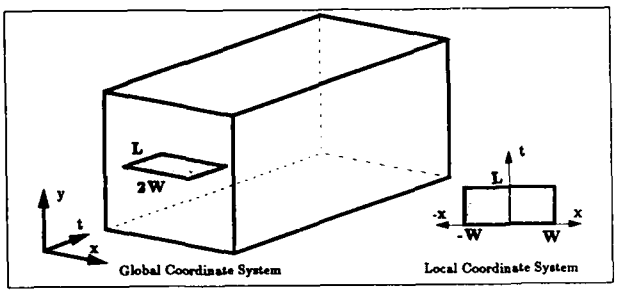



Figure 2: A *slice* from the sequence

the sequence at the same position, see fig. 2. It can be displayed as an image, the vertical and horizontal axes corresponding to the time and spatial directions respectively.

This spatio-temporal data structure provides an easy way to trace a line segment through frames. Assume there is an edgel **P** on a line segment under translation $\vec{V}$ in frame $i$, it moves to $\mathbf{P}'$ in frame $j$. If we construct a slice centered at **P** with inclination $\theta$, the 1-D image in the $j$th frame picks up another point $\mathbf{P}''$ because in general the orientation of the slice is different from that of the translation $\vec{V}$, see fig. 3.

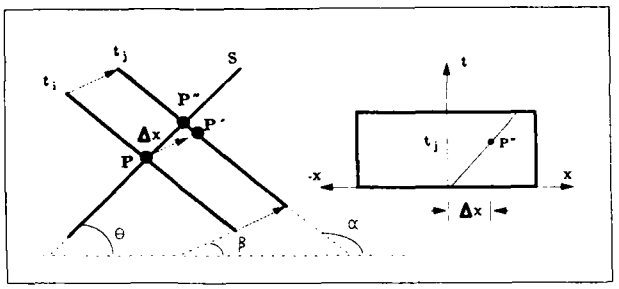



Figure 3: A *path* in a *slice*

Since we assume high sampling rate, there are points between **P** and $\mathbf{P}''$ corresponding to the line segment in frames in between, the sequence of those points is called the *path* of **P** in the *slice* . The slope of the path gives an estimate of the speed form **P** to $\mathbf{P}''$,

$$V_\theta = \frac{\mathbf{P}'' - \mathbf{P}}{t_j - t_i} = \frac{\Delta x}{\Delta t}$$

The longer the path can be traced, the better the estimate of $V_\theta$. Therefore all the $V_\theta$ estimates are associated with a *confidence factor* proportional to the length of the corresponding path.

$V_\theta$ alone is not enough to determine the real velocity $\vec{V}$, it only provides a constraint that the projection of $\vec{V}$ onto the normal of the normal of the line segment should be the same as the component of $V_\theta$ along the normal direction. Let $\alpha$ be the inclination of the line segment and $\beta$ be the orientation of $\vec{V}$, then we have the following relation

$$V_\theta = \frac{\|\vec{V}\| \cdot \sin(\alpha - \beta)}{\sin(\alpha - \theta)}$$

The orientation of the 1-D image, $\theta$, may be arbitrary, we choose the most convenient four orientations: $-45°$, $0°$, $45°$ and $90°$. The corresponding *slices* are called $S_{-45}$, $S_0$, $S_{45}$ and $S_{90}$, see fig. 4.

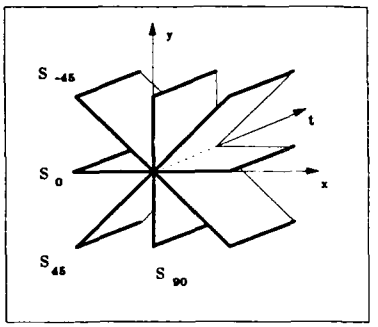



Figure 4: Different orientations of *slices*

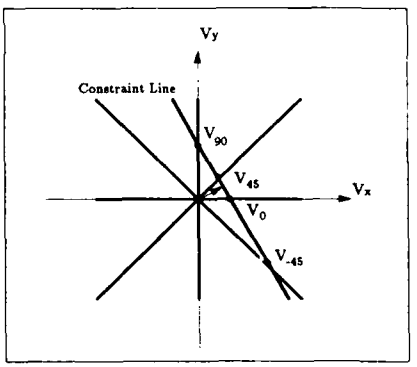



Figure 5: The normal velocity and *constraint line* in velocity space

For each edge point detected by the Canny edge detector [5], four *slices* are constructed with all the 1-D images form frame 0 to frame **L - 1** centered at the position of the edgel in frame 0. The velocities estimated from the *slices* fall on a line in the velocity space, see fig. 5. We can simply fit a line to the velocity points based on least square error weighted by the *confidence factor*, and find the perpendicular vector from origin to the line. The perpendicular vector is the normal velocity $\vec{N}$ and the fitted line is called the *constraint line*. Although two slices are good enough to determine the *constraint line*, we use four to reduce the chance of alignment in a digitized process.

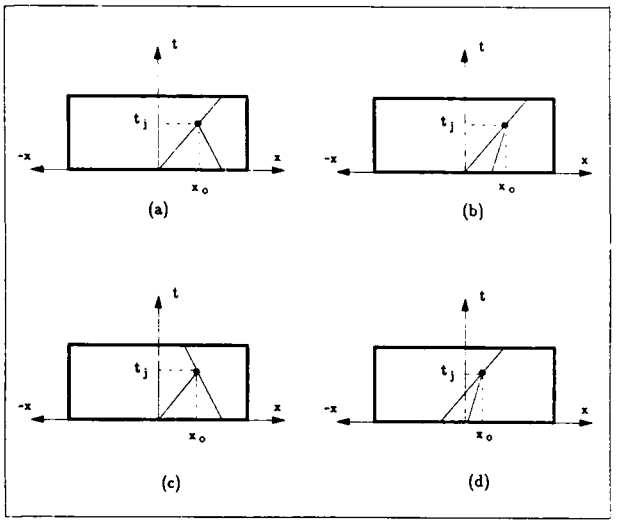



Figure 6: *Paths* with Occlusion

Besides the slope, the topology of *paths* in a *slice* also gives important information. In fig. 6 (a), the line segment to which the edgel **P** initially belongs, will occlude some other line segment $x_o$ unit length away from **P** along the direction $\theta$ at time $t_j$. The Same message is carried in figure 6 (b) except that the two lines are moving in opposite direction in (a) while both are moving in the same direction at different speed in (b). Figure 6 (c) and (d) show that **P** is on a line segment about to be occluded.

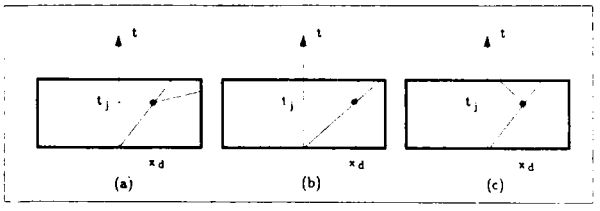



Figure 7: *Paths* with Disocclusion

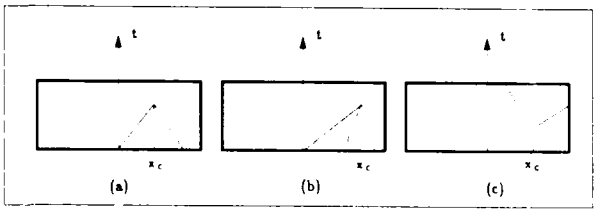



Figure 8: *Slice* with Corner

Figure 7 shows the cases of disocclusion, in which a new line segment shows up at the position $x_d$ unit length away from **P** along the direction $\theta$ in the $j$th frame. The new line segment is slower than the current one in (a), faster in (b) and moving in a different direction in (c). Figure 8 shows some examples when a corner is encountered. Corners are worth noticing because they can give both velocity components of the motion.

## 3.2 Segmentation Based on Motion Only

Once we have the normal flows assigned to the edge points in a frame, the next step toward image understanding is the *interpretation* of the flow field, which can subdivided into two stages, the first of these is to segment the edge points into contours and the second stage is to find the real velocities of the objects whose boundaries and surface markings give rise to those contours.

To segment edge points in an image frame without any *a priori* knowledge, problems may occur when there are more than one objects moving and occlusion and/or disocclusion take place. If one object is moving in front of another object then edge points on the boundaries of the rear surface will either be occluded or disoccluded during this movement, depending on whether the front object is moving to cover or uncover the object behind it.

The contours close to where the occlusion or disocclusion takes place will always form a three-way junction, where 2 branches belong to the front object while the third belongs to the rear one. One image frame along is not enough to tell which two branches go together. When we process the slices as mentioned in the previous section, the particular $Y$ or $\lambda$ shape paths will be noticed. Therefore, we can predict *where* and *when* the occlusions or disocclusions will happen, and send messages to the image frames to mark the places to watch out for occlusion or disocclusion.

Each frame will receive several messages from earlier image frame if there exists occlusion or disocclusion in the frame. Besides the location, the messages also give the *dominant velocities* within the spots of occlusion or disocclusion. The *dominant velocities* is the velocity of the motion of the front object along the inclination of the slice, i.e. the slope of the crossing path which terminates the other.

Now the segmentation becomes easier because the ambiguity of the three-way junctions is resolved. Whenever we trace a contour and get into an occlusion or disocclusion spot, we can use the similarities among the three branches to the *dominant velocities* to determine whether to break or extend the contour. Our segmentation program tries to generate contours as long as possible.

## 3.3 Computing the Correct Velocity Field along a Contour

The segmented contours are associated with the normal

flow estimates of each point. An additional constraint is used for the integration of local motion measurement to compute the two-dimensional velocity filed: the constraint is a smoothness constraint to minimize the variation of the velocity measurement along the contours, because the velocity field across a physical surface is generally expected to be smooth. The velocity field of least variation is in general not the physical correct one, however it is often qualitatively similar to the true velocity field. When the two velocity fields differ significantly, it appears that the smoothest velocity field may be more consistent with human motion perception [11]. The particular measure of variation we choose is $\int_C \left|\frac{\partial \vec{V}}{\partial s}\right|^2 ds$ : the integral of the square of velocity change along the contour.

If there exists at least two edge points at which the local orientation of the contour is different, then there exists an unique velocity field that satisfies the known normal velocities and minimize $\int_C \left|\frac{\partial \vec{V}}{\partial s}\right|^2 ds$. Since we have only discrete points, the first in the design of the algorithm is to convert the continuous formulation into a discrete one.

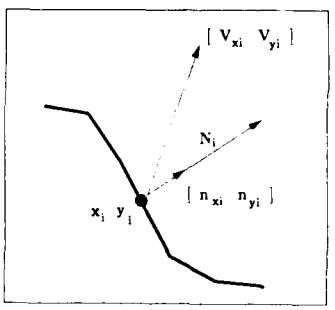



Figure 9: Illustration of the notations

Assume that the contour has $n$ edge points on it, { $(x_0, y_0)$, $(x_1, y_1)$, ..., $(x_{n-1}, y_{n-1})$ }. For each edge point, $(x_i, y_i)$, see fig. 9, we have an estimate of the normal velocity represented by the magnitude of the normal velocity, $N_i$, and the direction normal, $[n_{xi}, n_{yi}]$, perpendicular to the contour. We want to find a list of velocities, { $(Vx_0, Vy_0)$, $(Vx_1, Vy_1)$, ..., $(Vx_{n-1}, Vy_{n-1})$ } which minimize the variation,

$$\sum_{i=1}^{n-1} \left[(Vx_i - Vx_{i-1})^2 + (Vy_i - Vy_{i-1})^2\right] \qquad (2)$$

and satisfies the constraint that the component of the velocity in the normal direction equals the estimated normal velocity,

$$Vx_i \cdot n_{xi} + Vy_i \cdot n_{yi} = N_i, \quad \text{i} = 0 \ldots \text{n-1} \qquad (3)$$

To loosen the constraints, equation (3) does not have to be exactly satisfied. Therefore the energy function, $\phi$, is defined as a linear combination of the above two equations, which is

$$\phi = \sum_{i=1}^{n-1} \left[(Vx_i - Vx_{i-1})^2 + (Vy_i - Vy_{i-1})^2\right] + \gamma \cdot \sum_{i=0}^{n-1} [Vx_i \cdot n_{xi} + Vy_i \cdot n_{yi} - N_i]^2 \qquad (4)$$

To find out the set of velocities { $(Vx_0, Vy_0)$, $(Vx_1, Vy_1)$, ..., $(Vx_{n-1}, Vy_{n-1})$ } which minimizes the energy function in equation (4), one can take partial derivatives of $\phi$ with respect to $Vx_i$ and $Vy_i$, where

$$\begin{aligned} \frac{\partial \phi}{\partial Vx_i} &= 0, \quad \text{and} \\ \frac{\partial \phi}{\partial Vy_i} &= 0, \quad \text{for i = 0 ... n-1} \end{aligned} \qquad (5)$$

From equation 5 we have $2n$ linear equations for $2n$ unknowns. We can use any method to solve the linear system as long as not all the edge points are on a straight line. The method we choose is a *conjugate gradient algorithm*, which finds a solution in $2n$ iterations with the initial guess $Vx_i = N_i n_{xi}$ and $Vy_i = N_i n_{yi}$.

## 3.4 Higher Level Motion Processing

After the segmentation and variation minimization, we have a set of contours associated with velocity estimate along them to represent the optical flow field in the dynamic scene. A great deal of information could be picked up from the flow field even without invocation of high level processing like object recognition. For example, the flow field is rich enough to support the inference of collision when a robot is moving in an unknown place, or to locate the *focus of expansion* for navigation.

The contours also outline the surface patches. With the velocities of the surfaces and their spatial relations in the two-dimensional scene, their three-dimensional structures and the three-dimensional motion may be determined.

In particular, it should follow that, away from the boundaries, adjacent pixels should have similar motion, pixels corresponding to the same physical location should have similar intensities, and the resulting path should be smooth [23]. Therefore the velocities assigned to the contours can be propagated into the *interior* of the surface, using Nagel's formulation [19] for instance, to generate a dense velocity field. We still have to make sure that the contour segments correspond to correct object bound-

aries. Otherwise the velocity fields might *bleed* into other irrelevant regions, which happens often if occlusion or disocclusion are not handled correctly.

# 4 Results

We have generated a sequence of synthetic images for the purpose of testing and illustration, in which there are two rectangles, see fig. 10. The longer axis of the leftmost rectangle makes a 30° with the x axis. It is moving in the north east direction half a pixel per frame, while the right rectangle is moving in the north west direction with the same speed. The first and twelfth frames are shown in fig. 10 (a) and (b). A typical example of the slice analysis is shown in fig. 11, in which an edge point on the lower left boundary of the left rectangle in the fourth frame is processed. The four slices are in the upper right pane while the right hand side shows the result of edge detection on the slices. The lower right pane shows the velocity points in the velocity space and the *constraint line* fitted to them, and the normal velocity assigned to the edge point. Figure 12 shows the normal flow by collecting all the estimates of normal velocity for edge points in the fourth frame. Figure 13 shows the result after segmentation and variation minimization. The fourth frame got several messages passed from previous frames and located the places of occlusion so that physically related edge points are grouped together. In this example, there are only two contours and the variation minimization algorithm is applied to both of them separately with $\gamma = 0.01$. The constructed velocity field is very close to the real value both quantitatively and qualitatively.

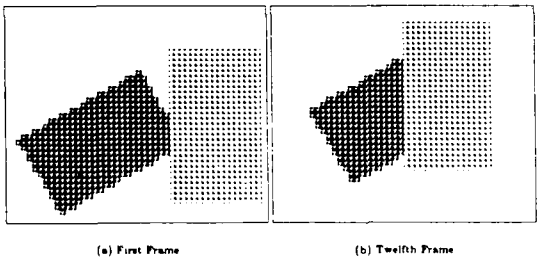



Figure 10: Synthetic Sequence of Two Rectangles

The next example is a real image sequences from SRI[2], in which the camera is moving forward in a lab. We only use a 32 by 32 window containing some palm leaves, since a smail window is already good enough to demonstrate this local process. The first frame of the sequence with the window boundary highlighted is shown in fig. 14.

[2]Courtesy of Dr. Baker and Dr. Bolles

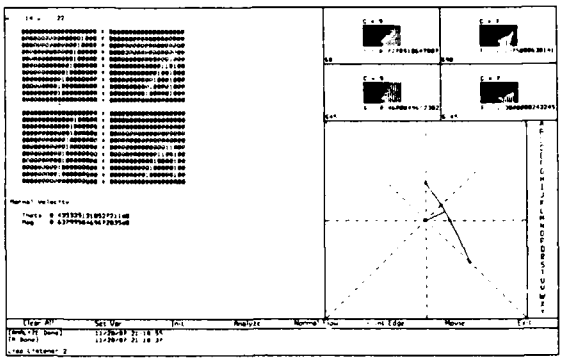


Figure 11: Slice Analysis

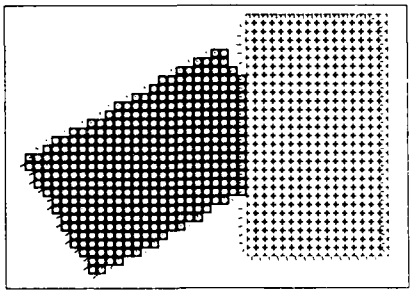


Figure 12: Normal Flow Field of the Fourth Frame

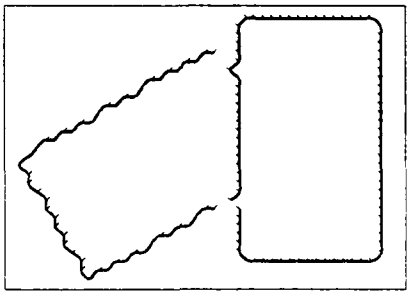


Figure 13: Segmented Contours and Velocity Field with $\gamma = 0.01$

The final result of segmentation and variation minimization is shown in fig. 15. Since the camera is pointing at some point to the right of the window while it moves, the sequence looks like the leaves are moving towards the viewer and passing by his left hand side with a little expansion. The last example is an image sequence taken by a robot while it is moving down a hallway. We choose a 64 by 64 window of the scene containing a chair against the wall, see fig. 16. The segmented contours and variation minimized velocity flow is shown in figure 17.

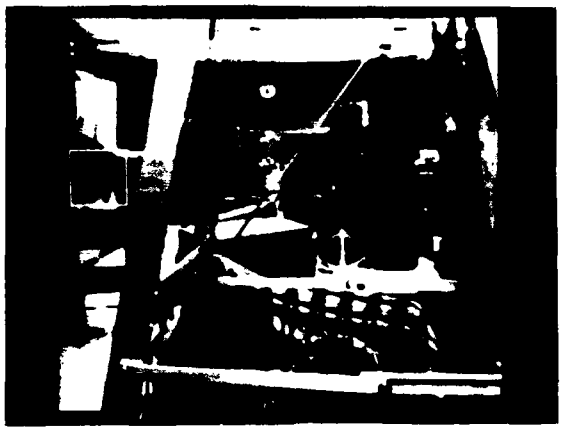


Figure 14: SRI Sequence: Zoom

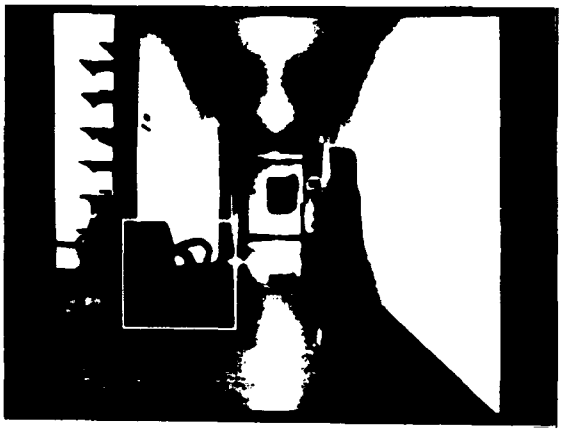


Figure 16: SRI Sequence: Hallway

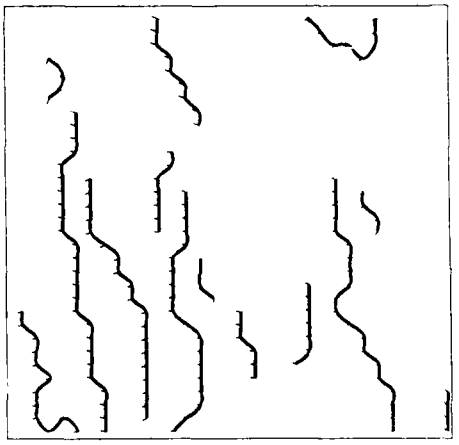


Figure 15: Segmented Contours and Velocity Field with $\gamma = 0.01$

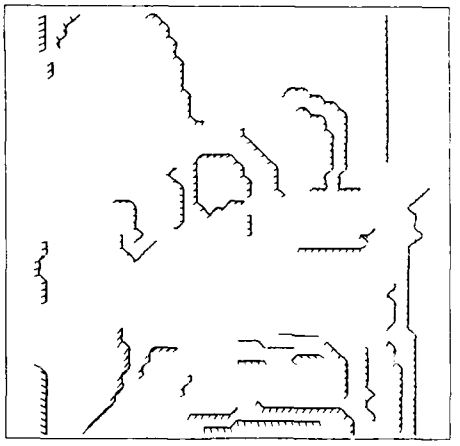


Figure 17: Segmented Contours and Velocity Field with $\gamma = 0.01$

## 5 Conclusion

We have presented a spatio-temporal approach to solve the early processing problem of motion analysis, which can handle scenes of multiple moving objects with occlusion and/or disocclusion. The characteristics and advantages of this method are as follows:

1. *Parallel Processing in the Spatial Domain*
   the slice analyses of the edge points are independent so they can proceed in parallel. Assuming that there is a 2-D SIMD array as large as the image with each entry of the array an identical processing element, each processing element has a memory **L** words long and is able to talk to its neighbors **W** steps away in eight directions. All the processing elements can construct their slices at the same time via local communication and analyze the slices concurrently. This property promises a hardware implementation with identical units performing the same operations in parallel.
2. *Pipeline Processing in the time dimension*
   the slice analysis only takes **L** frames. Each time we pump in a new image frame, it is distributed over

the entire array and each processing element simply fetches the corresponding pixel and gets rid of the oldest one in its local memory. While the array is working on segmentation and variation minimization of the current frame, the slice analysis can be invoked to work on the next frame.

3. *Symbolic scene description*
   the segmentation of moving objects are generated frame by frame, in which the contours provide the skeleton of the scene while the velocity field gives their relations over frames. Therefore, our process extracts suitable information for higher level interpretation process.
4. *Occlusion and Disocclution*
   our method is able to identify occlusion and/or disocclusion, which is very important to circumvent ambiguity.
5. *Incremental Process*
   Our method can be extended such that after having processed a few frames, the system is able to predict the path in a slice. The predicted value can either reduce computation time or lead to better estimation.

Finally, with those benefits, one more point the authors would like to state is the liberal assumptions of this work. The assumptions are few and yet need to be loosely satisfied. Although the high sampling rate assumption allows us to approximate any short trajectory of movement by translation and approximate any portion of the object boundary by piecewise straight line, we want to extend this work to handle rotation and motion at varying speeds. The only change is to allow curved paths in a slice instead of purely straight lines. The only ambiguity is that the paths for motion at changing speed and for curved object boundary are both curved. More work is also required to study the effects of quantization and noise sensitivity in a larger number of real image sequences.

# References


[1] P. Anandan, " A Unified Perspective on Computational Techniques for Measurement of Visual Motion," *Proceedings of Image Understanding Workshop*, Vol. 2, February 1987, pp. 719-732.

[2] M. Asada, M. Yachida, and S. Tsuji, "Three dimensional Motion Interpretation for the Sequences of Line Drawings," *Proceedings of the International Conference on Pattern Recognition*, 1980, pp. 1266-1273.

[3] S. T. Barnard and W. B. Thompson, "Disparity Analysis of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 2, No. 4, July 1980, pp. 333-340.

[4] R. C. Bolles and H. H. Baker, "Epipolar-Plane Image Analysis: A Technique for Analyzing Motion Sequences," *Proceedings of the Third Workshop on Computer Vision: Representation and Control*, Bellaire, Michigan, October 1985, pp. 168-178.

[5] J. Canny, " A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, November 1986, pp. 679-698.

[6] L. S. Davis, Z. Wu, and H. Sun, "Contour-Based Motion estimation," *Computer Vision, Graphics and Image Processing*, Vol. 23, 1983, pp. 313-326.

[7] L. Dreschler and H.-H. Nagel, "Volumetric Model and 3-D Trajectory of a Moving Car Derived from Monocular TV-frame Sequence of a Street Scene," *International Joint Conference on Artificial Intelligence*, Vancouver, Canada, August 1981.

[8] C. L. Fennema and W. B. Thompson, "Velocity Determination in Scenes Containing Several Moving Objects," *Computer Graphics and Image Processing*, Vol. 9, April 1979, pp. 301-315.

[9] D. B. Genery, "Object Detection and Measurement Using Stereo Vision," *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, August 1979.

[10] F. Glazer, G. Reynolds and P. Anandan, "Scene Matching by Hierarchical Correlation," *Proceedings of Computer Vision and Pattern Recognition*, June 1983, pp.432-441.

[11] E. C. Hildreth, "The Computation of the Velocity Field," Technical report 734, Massachusetts Institute of Technology, September 1983.

[12] C. J. Jacobus, R. T. Chien and J. M. Selander, "Motion Detection and Analysis by Matching Graphs of Intermediate Level Primitives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 2, No. 6, November 1980, pp. 495-510.

[13] R. Jain and H.-H. Nagel, "On the Analysis of Accumulative Difference Pictures from Image Sequences of Real World scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 1, No. 2, April 1979, pp. 206-214.

[14] R. Jain, "Extraction of Motion Information from Peripheral Processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 3, September 1981, pp. 489-503.

[15] S. Y. Kung, "On Supercomputing with Systolic / wavefront Array Processors," *IEEE Proceedings*, Vol. 72, No. 7, July 1984, pp. 867-884.

[16] D. H. Marimont, "Projective Duality and the Analysis of Image Sequences," *Proceeding of IEEE Workshop on Motion: Representation and Analysis*, May, 1986, pp. 7-14.

[17] G. Medioni and R. Nevatia, "Segment-Based Stereo Matching," *Computer Vision, Graphics, and Image Processing*, Vol. 31, 1985, pp. 2-18.

[18] H. P. Moravec, *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*, Ph. D. dissertation, Stanford University, 1980, also available as Carnegie-Mellon University RI-RT-3, Robotics Institute.

[19] H.-H. Nagel and W. Enkelmann, "An investigation of Smoothness Constraints for the Estimation of Displacement Vector Fields from Image Sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 5, September 1986, pp. 565-593.

[20] J. M. Prager, "Segmentation of Static and Dynamic Scenes," Technical Report TR 79-7, Dept. of Computer and Information Science, University of Massachusetts, May 1979.

[21] K. Price and R. Reddy, "Matching Segments of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 1, No. 1, January 1979, pp. 110-116.

[22] V. S. Ramachandran and S. M. Anstis, "The Perception of Apparent Motion," *Scientific American*, June, 1986, pp. 102-109.

[23] I. K. Sethi and R. Jain, "Finding Trajectories of Feature Points in a Monocular Image Sequence," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 1, Jan. 1987, pp. 56-73.

[24] M. A. Shah and R. Jain, "Detecting Time Varying Corners," *Computer Vision, Graphics, and Image Processing*, Vol. 28, 1984, pp. 345-355

[25] W. E. Snyder, "Computer Analysis of Time Varying Images," *Computer*, Vol. 14, No. 8, August 1981, pp. 7-9.

[26] J. Tanner and C. Mead, "An Integrated Analog Optical Motion sensor," *IEEE ASSP Society Workshop on VLSI Signal Processing, II* November, 1986, pp. 59-76.

[27] T. D. Williams, "Depth of Camera Motion in a Real World Scene," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 2, No. 6, November 1980, pp. 511-516.

# NATURAL REPRESENTATION OF MOTION IN SPACE-TIME

Wolfgang O. Franzen

Institute for Robotics and Intelligent Systems
School of Engineering
University of Southern California
Los Angeles, California 90089-0273

## Abstract

This paper describes a new approach to the representation of rigid and nonrigid 3D motion in time-varying imagery. The homogeneous coordinate representation of rigid transformations is extended by adding time as an explicit component. We characterize the classes of motion and/or structural change that have a constant matrix representation under this new notation. The coefficients of this matrix representation have a natural interpretation as the constant coefficients of a certain nonhomogeneous system of linear difference equations.

Among the classes of object motion having a constant representation are the following: a "constantly deforming" object undergoing translation; a rigid object rotating about a fixed axis at a constant angular velocity while undergoing translation in any direction and acceleration in the direction of the axis of rotation; a nonrotating rigid object undergoing constant translation and/or acceleration in any direction.

Camera motion, object motion, or simultaneous camera and object motion, may all be described using this representation. However, simultaneous camera and object motion is *not* equivalent to either object motion alone, or to camera motion alone. We derive a vector equation which expresses the position of a point on an object, at an arbitrary juncture in time, as a function of its initial position, and the matrices describing the motion of the camera, and the motion of the object.

The matrix representation presented in this paper is compact and should allow the efficient computation of the motion of computer generated objects. It is straightforward to calculate the matrix representation given the underlying motion parameters, and vice versa.

As an application of the methodology developed in this paper, we present the following novel result. If a rigid object, undergoing constant motion of the form described in this paper, is accelerating solely due to a constant external force of known magnitude, we show how the *absolute distances* to points on the object, and the absolute parameters of motion, may be recovered from a monocular image sequence.

## 1 Introduction

The analysis of motion in time-varying imagery is an active research area in computer vision. There are a number of good surveys on the subject, such as [Nag86].

Over the last few years, there has been an increasing trend of imposing constraints in addition to rigidity, such as constancy of motion, to facilitate the analysis of motion, or to solve the structure from motion problem. In his now classic work, Ullman [Ull79] originally solved the structure from motion problem (orthograhic projection) for four points in three frames, using no assumption other than rigidity. The best recent work, that imposes additional contraints, is that of Shariat [Sha86], who studied objects undergoing uniform translation and rotation. Among other cases, he solved the structure from motion problem (perspective projection) using only three points in three frames.

The approach presented in this paper is somewhat different. Rather than explicitly putting contraints on motion, we start with systems of equations whose solution leads to nontrivial, but mathematically tractable, classes of motion. The proper selection of such a system is a matter of physical and mathematical intuition. Rigid motion of the form described in this paper is more general than the motion studied by Shariat, and often allows the solution of the structure from motion problem for the same number of points in the same number of frames, as in his case. What is more important, this representation also allows the study of *structure from nonrigid motion*. With the notable exception of [Che85], [HP86], [Sub86], and [CP86], relatively little work has been done on the quantitative analysis and representation of nonrigid motion.

In the following, we begin with a brief review of homogeneous coordinates. Then a generalization of homogeneous coordinates, that we call chronogeneous coordinates, is described. (The term chronogeneous is actually a contraction of *chrono*-homo*geneous*). After introducing some additional notation, we derive a vector equation that expresses the position of a point, at an arbitrary juncture in time, in terms of its initial position and the matrices describing the motion of the object and the motion of the camera. Then a characterization of chronogeneous motion is given, with particular emphasis on rigid motion. A novel result involving the recovery of absolute depth from a monocular image sequence is presented. Finally, we summarize what we believe to be the major contributions of this work, and discuss future research.

## 2 Homogeneous Coordinates

Rigid transformations of bodies are typically represented using homogeneous coordinates. Homogeneous coordinates were introduced by Roberts in [Rob68]. [Pau81] also provides a good overview of homogeneous transformations. The usefulness of this representation stems form the fact that rigid transformation and perspective are expressible in matrix form. The homogeneous coordinate representation of the 3D point $(x, y, z)^T$ is any 4D point of the form $(\omega x, \omega y, \omega z, \omega)^T$ where $\omega \neq 0$. The value of the last component, $\omega$, is normally taken to be 1, until a perspective projection operator is applied.

In the following, let $\vec{x}_{3D}$ be the 3D position of a point, and let $\vec{x}_{4D}$ be its corresponding homogeneous representation. Also, let $\vec{x}'_{3D}$ and $\vec{x}'_{4D}$ represent corresponding transformed positions of these points. A general homogeneous transformation may be expressed as

$$\vec{x}'_{4D} = \mathcal{H}\vec{x}_{4D}$$

where

$$\mathcal{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix}$$

is the homogeneous transformation matrix.

A general rigid 3D tranformation may be expressed as

$$\vec{x}'_{3D} = \mathcal{R}\vec{x}_{3D} + \vec{T}$$

where

$$\mathcal{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

is a rotation matrix, and $\vec{T} = (t_1, t_2, t_3)^T$ is a translation vector.

The same transformation is expressed more succinctly in homogeneous coordinates as

$$\vec{x}'_{4D} = \mathcal{H}\vec{x}_{4D}$$

where

$$\mathcal{H} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Despite the usefulness of homogeneous coordinates for representing a rigid transformation between two frames, this representation does not lend itself to describing motion in multiframe imagery. This is due to the fact that even for relatively simple kinds of motion, the homogeneous transformation matrix changes from one frame to the next. Some examples of simple types of motion that yield changing $\mathcal{H}$-matrices are:

- The ballistic motion of a (nonrotating) ball accelerating due to the force of gravity.
- The motion of a (nonrotating) camera on a uniformly accelerating vehicle.
- The motion of a spinning top that is rolling across the floor (assume no precession).
- The motion of a wheel of a car moving at a constant velocity, when viewed from the side.

Ideally, one would like to describe motion in such a way that the motion parameters corresponding to commonly occuring types of motion are constant over time. This motivates the following extension to homogeneous coordinates, which allows the natural description of the above types of motion, as well as other types, even nonrigid motion.

## 3 Chronogeneous Coordinates

This section describes a generalization of homogeneous coordinates in the time domain, which we call *chronogeneous coordinates*. The homogeneous coordinate representation is extended by augmenting it to encode time explicitly. The chronogeneous coordinate representation of the 3D point $(x, y, z)^T$ at time $t$ is any 5D point of the form $(\omega x, \omega y, \omega z, t, \omega)^T$ where $\omega \neq 0$. The value of the last component, $\omega$, is normally taken to be 1, until a perspective projection operator is applied. Note that the factor $\omega$ does not multiply the time component. Whereas the spatial components of a point, at least conceptually, range over a continuous set of values, the time component is discrete and only takes on values which are multiples of $\Delta T$, where $1/\Delta T$ is the frame rate of the imaging system. The frame rate is assumed to be a known constant.

Except for the perspective projection matrix discussed below, we will consider only chronogeneous transformation matrices of the following form, which we call *standard* chronogeneous matrices:

$$\mathcal{C} = \begin{bmatrix} s_{11} & s_{12} & s_{13} & \gamma_1 & p_1 \\ s_{21} & s_{22} & s_{23} & \gamma_2 & p_2 \\ s_{31} & s_{32} & s_{33} & \gamma_3 & p_3 \\ 0 & 0 & 0 & 1 & \delta t \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{S}} & \vec{\Gamma} & \vec{P} \\ \hline 0 & 0 & 0 & 1 & \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \quad (1)$$

Since almost all the chronogeneous matrices we discuss are standard, we often drop this designation.

The value of the element $\delta t$, of $\mathcal{C}$, is restricted to being an integer multiple of $\Delta T$. In fact, the value will always be a known multiple of $\Delta T$. Therefore, this representation has 15 degrees of freedom and, in general, represents nonrigid motion. We sometimes refer to the submatrix, $\mathcal{S}$, of $\mathcal{C}$, as the *structural deformation submatrix*, or simply the *deformation submatrix*. If $\mathcal{S}$ is a rotation matrix, then $\mathcal{C}$ represents a rigid transformation and has only 9 degrees of freedom. The subvector $\vec{\Gamma} = (\gamma_1, \gamma_2, \gamma_3)^T$ has units of velocity, but roughly encodes information about acceleration, and the subvector $\vec{P} = (p_1, p_2, p_3)^T$ has units of displacement, but roughly encodes information about velocity.

The matrix, $\mathcal{C}$, is equivalent to the following 3D transformation:

$$\vec{x}'_{3D}(t + \delta t) = \mathcal{S}\vec{x}_{3D}(t) + t\vec{\Gamma} + \vec{P} \quad (2)$$

In addition, it causes the time to be incremented by the amount $\delta t$. If $\mathcal{S}$ is a rotation matrix, and if we drop the dependence on time and view $t\vec{\Gamma} + \vec{P}$ as the translation vector, then the above

vector equation reduces to the general rigid 3D transformation. Vector equation (2) is a nonhomogeneous system of first order linear difference equations with constant coefficents. Therefore, the theory of linear difference equations may be used to study solutions of this equation.

# 4 Common Notation and Assumptions

This section defines some common notation and assumptions that are used throughout the remainder of the document.

## 4.1 Special Chronogeneous Matrices and Operators

This section introduces some often used chronogeneous matrices, as well as the perspective division operator. The $(5 \times 5)$ identity matrix is denoted by $\mathcal{I}_5$. Similarly, the $(3 \times 3)$ identity matrix is denoted by $\mathcal{I}_3$.

The following matrix, $\mathcal{T}$, leaves the spatial location of a point unaltered, but advances the time component by one interframe time interval:

$$\mathcal{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta T \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The following equation holds:

$$\mathcal{T} \begin{pmatrix} x \\ y \\ z \\ t \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ t + \Delta T \\ 1 \end{pmatrix} \tag{4}$$

We now define the perspective projection matrix, $\mathcal{P}$. In our work, we use *left-handed* coordinate systems. For the camera coordinate system, the z-axis points in such a way that positive distances are in front of the camera. For simplicity, and without loss of generality, we assume that *all* distances are measured in the same units (this includes image plane coordinates). Then, with the camera coordinate system centered on the camera lens center,

$$\mathcal{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 & 0 \end{bmatrix} \tag{5}$$

where $f$ is the focal length of the camera. Alternatively, if the camera coordinate system is centered on the image plane, then

$$\mathcal{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 & -1 \end{bmatrix} \tag{6}$$

The following operator, $\mathcal{D}$, is used in conjunction with $\mathcal{P}$ to define the image plane coordinates of a point in terms of its camera chronogeneous coordinates. The operator $\mathcal{D}$ is defined as

$$\mathcal{D}\left[\begin{pmatrix} x \\ y \\ z \\ t \\ \omega \end{pmatrix}\right] = \begin{pmatrix} x/\omega \\ y/\omega \end{pmatrix} \tag{7}$$

## 4.2 Coordinate Notations

Consider an image sequence, taken by a moving camera, consisting of "$n_f$" images (0 through $n_f - 1$), and consider a moving object having "$n_p$" points (0 through $n_p - 1$) that are visible in each of these images. We assume that both $\Delta T$, the interframe time interval, and $f$, the focal length of the camera, are known constants. Let $\vec{Q}_{i,j}$ refer to the position of the $j^{th}$ point at the $i^{th}$ discrete instant in time. We add a superscript to indicate in which reference frame and in what type of coordinates the position of the point is expressed.

- $\vec{Q}^c_{i,j}$ is a point expressed in chronogeneous coordinates in the camera reference frame.
- $\vec{Q}^s_{i,j}$ is the spatial (3D) part of the point $\vec{Q}^c_{i,j}$.
- $\vec{Q}^p_{i,j}$ is a point expressed in image plane 2D coordinates. It is the exact projection of the point $\vec{Q}^c_{i,j}$ onto the image plane.
- $\vec{Q}^m_{i,j}$ is the actually measured location of the point $\vec{Q}^p_{i,j}$ (in image plane 2D coordinates). It includes any "correspondence noise".

The following relationship expresses exact image plane coordinates in terms of camera chronogeneous coordinates:

$$\vec{Q}^p_{i,j} = \mathcal{D}[\mathcal{P}\vec{Q}^c_{i,j}] \tag{8}$$

where $\mathcal{P}$ is the perspective projection matrix, and $\mathcal{D}$ is the perspective division operator.

Assume that both the object and the camera are undergoing uniform chronogeneous motion. Let $\mathcal{A}$ be the (rigid) chronogeneous matrix that describes the motion of the camera. It describes a new camera position relative to the current instantaneous camera position. Let $\mathcal{B}$ be the chronogeneous matrix describing the motion of the object. Note that $\delta t_{\mathcal{A}} = \delta t_{\mathcal{B}} = \Delta T$.

# 5 Derivation of the "Coordinate Transformation Vector Equation"

The purpose of this section is to derive the coordinate transformation vector equation. This equation expresses the current chronogeneous position of a point in terms of its initial position, and the matrices describing the motion of the camera and the motion of the object. We first consider two subclasses of motion, and then derive the general coordinate transformation vector equation.

### 5.1 The Case of a Camera Moving Through a Static Environment

Consider a camera undergoing chronogeneous motion through a static environment. Motion of the camera has an inverse effect on object coordinates. Let us be more specific. Let us introduce the chronogeneous matrix $\mathcal{A}_R$ such that $\mathcal{A} = \mathcal{T}\mathcal{A}_R$, that is $\mathcal{A}_R = \mathcal{T}^{-1}\mathcal{A}$. The matrix $\mathcal{T}$ was defined previously, and causes time to advance by one "tick". The matrix $\mathcal{A}_R$ represents the spatial transformation that takes place between successive positions of the camera. Then the following relationships hold:

$$\mathcal{A}_R(\mathcal{T}^{-1}\vec{Q}^c_{i+1,j}) = \vec{Q}^c_{i,j}$$

and

$$\vec{Q}^c_{i+1,j} = \mathcal{T}\mathcal{A}_R{}^{-1}\vec{Q}^c_{i,j} = \mathcal{T}(\mathcal{T}^{-1}\mathcal{A})^{-1}\vec{Q}^c_{i,j} = \mathcal{T}\mathcal{A}^{-1}\mathcal{T}\vec{Q}^c_{i,j}$$

and therefore

$$\vec{Q}^c_{i,j} = (\mathcal{T}\mathcal{A}^{-1}\mathcal{T})^i\vec{Q}^c_{0,j} \tag{9}$$

If the camera is stationary, then $\mathcal{A}_R = \mathcal{I}_5$, $\mathcal{A} = \mathcal{T}$, and $\vec{Q}^c_{i,j} = \mathcal{T}^i\vec{Q}^c_{0,j}$.

### 5.2 The Case of a Stationary Camera Viewing a Moving Object

Consider a stationary camera viewing an object that is undergoing constant chronogeneous motion. The new camera chronogeneous coordinates of a point on the object are simply obtained by multiplying the current coordinates by $\mathcal{B}$, that is:

$$\vec{Q}^c_{i+1,j} = \mathcal{B}\vec{Q}^c_{i,j}$$

and in general

$$\vec{Q}^c_{i,j} = \mathcal{B}^i\vec{Q}^c_{0,j} \tag{10}$$

If the object is stationary, then $\mathcal{B} = \mathcal{T}$, and $\vec{Q}^c_{i,j} = \mathcal{T}^i\vec{Q}^c_{0,j}$.

### 5.3 Simultaneous Camera and Object Motion

After considering the previous two cases, derivation of the coordinate transformation vector equation is straightforward. If the camera were stationary, then the chronogeneous position corresponding to the $i^{th}$ image of a point would be $\mathcal{B}^i\vec{Q}^c_{0,j}$. If we consider only the spatial transformation involved, then the new position of the point due to object motion is $\mathcal{T}^{-1}\mathcal{B}^i\vec{Q}^c_{0,j}$. However, this point is viewed by a camera that has undergone motion. Therefore the composite effect is given by:

$$\vec{Q}^c_{i,j} = (\mathcal{T}\mathcal{A}^{-1}\mathcal{T})^i\mathcal{T}^{-1}\mathcal{B}^i\vec{Q}^c_{0,j} \tag{11}$$

The matrices in equation (11), the coordinate transformation vector equation, do not commute, and so the equation cannot be simplified. The implication of this vector equation is that, in general, simultaneous camera and object motion is not correctly modeled by either camera motion alone or object motion alone. This is due to the fact that, in general, there is no matrix, $\mathcal{C}$, such that $\mathcal{C}^i = (\mathcal{T}\mathcal{A}^{-1}\mathcal{T})^i\mathcal{T}^{-1}\mathcal{B}^i$. However, for every pure camera motion there is a pure object motion that has an identical effect on coordinate positions, and vice versa.

## 6 A Characterization of Constant Chronogeneous Motion

This section gives a characterization of the classes of motion that are representable by constant coefficient standard chronogeneous matrices. The following subsections discuss specific subcases in more detail. In each case we show how the components of the chronogeneous matrix are determined by the underlying parameters of motion, and how the motion parameters may be computed, given a chronogeneous matrix. At the end of this section, we briefly touch on the structure from chronogeneous motion problem.

Consider an arbitrary constant coefficient standard chronogeneous matrix, with deformation submatrix $\mathcal{S}$. This matrix represents the motion of some object, which is translating through space and structurally deforming according to the matrix $\mathcal{S}$. In addition, if the matrix expression $(\mathcal{I}_3 - \mathcal{S})$ is singular, the object may also be accelerating in a direction orthogonal to the subspace (of 3-space) spanned by $(\mathcal{I}_3 - \mathcal{S})$.

To make the foregoing more concrete, consider the case of a rigid object. In this case, the deformation submatrix $\mathcal{S}$ is a rotation matrix, call it $\mathcal{R}$. The matrix expression $(\mathcal{I}_3 - \mathcal{R})$ is singular. This is easily seen as follows. Let $\vec{A}$ be the (unit length) axis of rotation vector associated with $\mathcal{R}$. Then

$$(\mathcal{I}_3 - \mathcal{R})\vec{A} = \vec{A} - \mathcal{R}\vec{A} = \vec{A} - \vec{A} = \vec{0}$$

As the null space of $(\mathcal{I}_3 - \mathcal{R})$ contains a nonzero vector, this matrix expression is singular. Although we do not prove it here, $(\mathcal{I}_3 - \mathcal{R})$ is actually of rank 2, unless $\mathcal{R} = \mathcal{I}_3$. Therefore, if $\mathcal{R} \neq \mathcal{I}_3$, $\vec{A}$ spans the nullspace of $(\mathcal{I}_3 - \mathcal{R})$, and the general case of rigid chronogeneous motion corresponds to a rigid object rotating with fixed angular velocity, translating through space, and accelerating in the direction of the axis of rotation.

Figure 1 gives a taxonomy of the classes of rigid chronogeneous motion. These are discussed in the remainder of this section, after a discussion of the case of general deformation, with $(\mathcal{I}_3 - \mathcal{S})$ nonsingular.

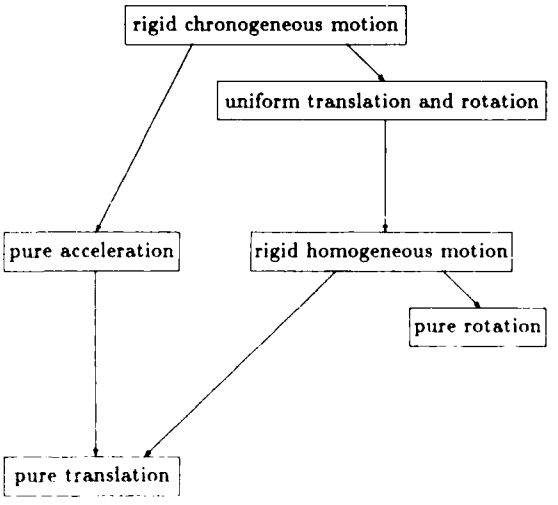



Figure 1: Taxonomy of Rigid Chronogeneous Motion

## 6.1 A Translating Deforming Object

Consider an object undergoing "constant deformation" about a center of deformation that is undergoing pure translation. Let $\vec{c}_i$ be the position of the center of deformation at the $i^{th}$ discrete instant in time. Then the following equation recursively determines the 3D position of a given point on the object, for a given deformation matrix S:

$$(\vec{Q}^s_{i+1,j} - \vec{c}_{i+1}) = \mathcal{S}(\vec{Q}^s_{i,j} - \vec{c}_i)$$

For a translating object, $\vec{c}_i = \vec{c}_0 + t\vec{V} = \vec{c}_0 + i\vec{V}\Delta T$, for some initial center of deformation, $\vec{c}_0$, and velocity vector, $\vec{V}$. The above equation may then be rewritten as follows:

$$\begin{aligned}
\vec{Q}^s_{i+1,j} &= \mathcal{S}(\vec{Q}^s_{i,j} - \vec{c}_i) + \vec{c}_{i+1} \\
&= \mathcal{S}(\vec{Q}^s_{i,j} - (\vec{c}_0 + i\vec{V}\Delta T)) + (\vec{c}_0 + (i+1)\vec{V}\Delta T) \\
&= \mathcal{S}\vec{Q}^s_{i,j} - \mathcal{S}\vec{c}_0 - i\mathcal{S}\vec{V}\Delta T + \vec{c}_0 + i\vec{V}\Delta T + \vec{V}\Delta T \\
&= \mathcal{S}\vec{Q}^s_{i,j} + i(\mathcal{I}_3 - \mathcal{S})\vec{V}\Delta T + (\mathcal{I}_3 - \mathcal{S})\vec{c}_0 + \vec{V}\Delta T \\
&= \mathcal{S}\vec{Q}^s_{i,j} + t(\mathcal{I}_3 - \mathcal{S})\vec{V} + (\mathcal{I}_3 - \mathcal{S})\vec{c}_0 + \vec{V}\Delta T
\end{aligned}$$

The chronogeneous matrix representing the same motion is:

$$\mathcal{B} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{S}} & (\mathcal{I}_3 - \mathcal{S})\vec{V} & (\mathcal{I}_3 - \mathcal{S})\vec{c}_0 + \vec{V}\Delta T \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \qquad (12)$$

We assume $(\mathcal{I}_3 - \mathcal{S})$ is nonsingular. Then given an arbitrary matrix

$$\mathcal{B} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{S}} & \vec{\Gamma} & \vec{P} \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

we may compute the parameters of motion, $\vec{V}$ and $\vec{c}_0$, as follows:

$$\vec{V} = (\mathcal{I}_3 - \mathcal{S})^{-1}\vec{\Gamma} \qquad (13)$$

$$\vec{c}_0 = (\mathcal{I}_3 - \mathcal{S})^{-1}(\vec{P} - \vec{V}\Delta T) \qquad (14)$$

## 6.2 Rigid Chronogeneous Motion: General Case

Consider the case where the deformation is actually a rigid rotation, $\mathcal{R}$. Assume $\mathcal{R} \neq \mathcal{I}_3$. The case $\mathcal{R} = \mathcal{I}_3$ corresponds to pure acceleration, and is treated elsewhere.

As discussed in the introductory remarks to this section, the general case of rigid chronogeneous motion corresponds to a rigid object rotating with fixed angular velocity, translating through space, and accelerating in the direction of the axis of rotation. The following recursive relationship holds

$$(\vec{Q}^s_{i+1,j} - \vec{c}_{i+1}) = \mathcal{R}(\vec{Q}^s_{i,j} - \vec{c}_i),$$

where we may write

$$\vec{c}_i = \vec{c}_0 + t\vec{V} + \tfrac{1}{2}\gamma t^2\vec{A} \quad \vec{c}_0 + i\vec{V}\Delta T + \tfrac{1}{2}i^2\gamma\vec{A}(\Delta T)^2$$

for some initial center of rotation, $\vec{c}_0$, initial velocity vector, $\vec{V}$, and signed magnitude of acceleration, $\gamma$. The axis of rotation, $\vec{A}$, is determined by $\mathcal{R}$. Substituting the formula for the position of the center of rotation into the recursive relationship, and simplifying, we obtain:

$$\begin{aligned}
\vec{Q}^s_{i+1,j} &= \mathcal{R}\vec{Q}^s_{i,j} + t((\mathcal{I}_3 - \mathcal{R})\vec{V} - \gamma\vec{A}\Delta T) \\
&\quad + (\mathcal{I}_3 - \mathcal{R})\vec{c}_0 + \vec{V}\Delta T - \tfrac{1}{2}\gamma\vec{A}(\Delta T)^2
\end{aligned}$$

The chronogeneous matrix representing the same motion is:

$$\mathcal{B} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{R}} & (\mathcal{I}_3 - \mathcal{R})\vec{V} - \gamma\vec{A}\Delta T & (\mathcal{I}_3 - \mathcal{R})c_0 + \vec{V}\Delta T - \tfrac{1}{2}\gamma\vec{A}(\Delta T)^2 \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \qquad (15)$$

We now derive expressions for the motion parameters, $\vec{c}_0$, $\vec{V}$, and $\gamma$, in terms of $\mathcal{R}$, $\vec{A}$, $\vec{\Gamma}$, $\vec{P}$, and $\Delta T$. The following relationships express the vector components of $\mathcal{B}$ in terms of the motion parameters:

$$\vec{\Gamma} = (\mathcal{I}_3 - \mathcal{R})\vec{V} - \gamma\vec{A}\Delta T \qquad (16)$$

$$\vec{P} = (\mathcal{I}_3 - \mathcal{R})\vec{c}_0 + \vec{V}\Delta T - \tfrac{1}{2}\gamma\vec{A}(\Delta T)^2 \qquad (17)$$

We first derive an expression for $\gamma$. From equation (16):

$$\begin{aligned}
\vec{A}\cdot\vec{\Gamma} &= \vec{A}\cdot((\mathcal{I}_3 - \mathcal{R})\vec{V} - \gamma\vec{A}\Delta T) \\
&= \vec{A}\cdot((\mathcal{I}_3 - \mathcal{R})\vec{V}) - \gamma(\vec{A}\cdot\vec{A})\Delta T \\
&= 0 - \gamma\Delta T \\
&= -\gamma\Delta T
\end{aligned}$$

and therefore

$$\gamma = -(\vec{A}\cdot\vec{\Gamma})/\Delta T$$

In the above, the symbol "$\cdot$" indicates dot product.

Next, we derive an expression for $\vec{V}$. From equation (17):

$$\begin{aligned}
\vec{A}\cdot\vec{P} &= \vec{A}\cdot((\mathcal{I}_3 - \mathcal{R})\vec{c}_0 + \vec{V}\Delta T - \tfrac{1}{2}\gamma\vec{A}(\Delta T)^2) \\
&= \vec{A}\cdot((\mathcal{I}_3 - \mathcal{R})\vec{c}_0) + \vec{A}\cdot(\vec{V}\Delta T) - \tfrac{1}{2}\gamma(\vec{A}\cdot\vec{A})(\Delta T)^2 \\
&= 0 + (\vec{A}\cdot\vec{V})\Delta T - \tfrac{1}{2}(-(\vec{A}\cdot\vec{\Gamma})/\Delta T)(\Delta T)^2 \\
&= (\vec{A}\cdot\vec{V})\Delta T + \tfrac{1}{2}(\vec{A}\cdot\vec{\Gamma})\Delta T
\end{aligned}$$

and therefore

$$\begin{aligned}
\vec{A}\cdot\vec{V} &= ((\vec{A}\cdot\vec{P}) - \tfrac{1}{2}(\vec{A}\cdot\vec{\Gamma})\Delta T)/\Delta T \\
&= \vec{A}\cdot(\vec{P}/\Delta T - \tfrac{1}{2}\vec{\Gamma})
\end{aligned}$$

In the following, the symbol "+" used as an exponent denotes the pseudoinverse operation. Readers who are unfamiliar with the pseudoinverse are referred to [LH74]. The pseudoinverse is a generalization of the inverse that also applies when the matrix is not square, or not of full rank (as in the following). We make use of the fact that $(\mathcal{I}_3 - \mathcal{R})^+\vec{A} = \vec{0}$. The initial velocity vector, $\vec{V}$, is determined as follows:

$$\begin{aligned}
\vec{V} &= (\mathcal{I}_3 - \mathcal{R})^+(\mathcal{I}_3 - \mathcal{R})\vec{V} + (\vec{A}\cdot\vec{V})\vec{A} \\
&= (\mathcal{I}_3 - \mathcal{R})^+(\vec{\Gamma} + \gamma\vec{A}\Delta T) + (\vec{A}\cdot\vec{V})\vec{A} \\
&= (\mathcal{I}_3 - \mathcal{R})^+\vec{\Gamma} + \gamma(\mathcal{I}_3 - \mathcal{R})^+\vec{A}\Delta T + (\vec{A}\cdot(\vec{P}/\Delta T - \tfrac{1}{2}\vec{\Gamma}))\vec{A} \\
&= (\mathcal{I}_3 - \mathcal{R})^+\vec{\Gamma} + (\vec{A}\cdot(\vec{P}/\Delta T - \tfrac{1}{2}\vec{\Gamma}))\vec{A}
\end{aligned}$$

Finally, we derive an expression for $\vec{c}_0$. The initial center of rotation, $\vec{c}_0$, is not uniquely determined. Adding any real multiple of $\vec{A}$ to $\vec{c}_0$ results in a physically indistinguishable motion. The derived value of $\vec{c}_0$ is the minimum length solution. From equation (17):

$$\begin{aligned}\vec{c}_0 &= (\mathcal{I}_3 - \mathcal{R})^+(\vec{P} - \vec{V}\Delta T + \tfrac{1}{2}\gamma\vec{A}(\Delta T)^2) \\ &= (\mathcal{I}_3 - \mathcal{R})^+(\vec{P} - \vec{V}\Delta T) \\ &= (\mathcal{I}_3 - \mathcal{R})^+(\vec{P} - ((\mathcal{I}_3 - \mathcal{R})^+\vec{\Gamma} + (\vec{A}\cdot(\vec{P}/\Delta T - \tfrac{1}{2}\vec{\Gamma}))\vec{A})\Delta T) \\ &= (\mathcal{I}_3 - \mathcal{R})^+(\vec{P} - (\mathcal{I}_3 - \mathcal{R})^+\vec{\Gamma}\Delta T - (\vec{A}\cdot(\vec{P}/\Delta T - \tfrac{1}{2}\vec{\Gamma}))\vec{A}\Delta T) \\ &= (\mathcal{I}_3 - \mathcal{R})^+(\vec{P} - (\mathcal{I}_3 - \mathcal{R})^+\vec{\Gamma}\Delta T)\end{aligned}$$

In summary, the motion parameters may be computed as follows:

$$\gamma = -(\vec{A}\cdot\vec{\Gamma})/\Delta T \tag{18}$$
$$\vec{V} = (\mathcal{I}_3 - \mathcal{R})^+\vec{\Gamma} + (\vec{A}\cdot(\vec{P}/\Delta T - \tfrac{1}{2}\vec{\Gamma}))\vec{A} \tag{19}$$
$$\vec{c}_0 = (\mathcal{I}_3 - \mathcal{R})^+(\vec{P} - (\mathcal{I}_3 - \mathcal{R})^+\vec{\Gamma}\Delta T) \tag{20}$$

### 6.3 Uniform Translation and Rotation

For this subclass of rigid chronogeneous motion, $\gamma = 0$. This class of motion has eight degrees of freedom. There are three degrees of rotational freedom, three degrees of freedom for the velocity vector, and two degrees of freedom for the center of rotation. The chronogeneous matrix representing this motion is:

$$\mathcal{B} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{R}} & (\mathcal{I}_3 - \mathcal{R})\vec{V} & (\mathcal{I}_3 - \mathcal{R})\vec{c}_0 + \vec{V}\Delta T \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{21}$$

### 6.4 Rigid Homogeneous Motion

This case corresponds to the class of motion representable by homogeneous transformations. $\vec{\Gamma} = \vec{0}$ in this case, and this class of motion has six degrees of freedom. For $\mathcal{R} \neq \mathcal{I}_3$, a general motion of this class consists of rotation, coupled with a restricted form of translation. Translation, if any, occurs in the direction of the axis of rotation. This may more commonly be described as helical or "barberpole" motion. The following recursive relationship holds

$$(\vec{Q}^s_{i+1,j} - \vec{c}_{i+1}) = \mathcal{R}(\vec{Q}^s_{i,j} - \vec{c}_i)$$

where

$$\vec{c}_i = \vec{c}_0 + t\nu\vec{A} = \vec{c}_0 + i\nu\vec{A}\Delta T$$

for some initial center of rotation, $\vec{c}_0$, and (signed) magnitude of velocity, $\nu$. The chronogeneous matrix representing this motion is:

$$\mathcal{B} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{R}} & \vec{0} & (\mathcal{I}_3 - \mathcal{R})\vec{c}_0 + \nu\vec{A}\Delta T \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{22}$$

Given a chronogeneous matrix of the above form, the motion parameters, $\nu$ and $\vec{c}_0$, may be computed as follows:

$$\nu = (\vec{A}\cdot\vec{P})/\Delta T \tag{23}$$
$$\vec{c}_0 = (\mathcal{I}_3 - \mathcal{R})^+\vec{P} \tag{24}$$

### 6.5 Pure Rotation

Pure rotation is a subclass of rigid homogeneous motion, with $\nu = 0$. This class of motion has five degrees of freedom. There are three degrees of rotational freedom, and two degrees of freedom for the center of rotation. The recursive relationship simplifies to

$$(\vec{Q}^s_{i+1,j} - \vec{c}_0) = \mathcal{R}(\vec{Q}^s_{i,j} - \vec{c}_0)$$

and the chronogeneous matrix corresponding to this motion is

$$\mathcal{B} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{R}} & \vec{0} & (\mathcal{I}_3 - \mathcal{R})\vec{c}_0 \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{25}$$

### 6.6 Pure Acceleration

For this subclass of rigid chronogeneous motion, $\mathcal{R} = \mathcal{I}_3$. This class of motion has six degrees of freedom. The following relationships hold:

$$\begin{aligned}\vec{Q}^s_{i,j} &= \vec{Q}^s_{0,j} + t\vec{V} - \tfrac{1}{2}\gamma t^2\vec{A} \\ &= \vec{Q}^s_{0,j} + i\vec{V}\Delta T - \tfrac{1}{2}i^2\gamma\vec{A}(\Delta T)^2\end{aligned}$$

for some initial velocity vector, $\vec{V}$, magnitude of acceleration, $\gamma$, and axis of *acceleration*, $\vec{A}$. When comparing this subclass of motion to the general case, we see that there is no reference to a center of rotation, and the axis of rotation has been replaced by an axis of acceleration, that is free to point in any direction. The recursive relationship for this motion is

$$\vec{Q}^s_{i+1,j} = \vec{Q}^s_{i,j} - t\gamma\vec{A}\Delta T + \vec{V}\Delta T - \tfrac{1}{2}\gamma\vec{A}(\Delta T)^2$$

and the chronogeneous matrix representing this motion is

$$\mathcal{B} = \left[\begin{array}{ccc|c|c} \multicolumn{3}{c|}{\mathcal{I}_3} & -\gamma\vec{A}\Delta T & \vec{V}\Delta T - \tfrac{1}{2}\gamma\vec{A}(\Delta T)^2 \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{26}$$

Given a chronogeneous matrix, with $\mathcal{S} = \mathcal{R} = \mathcal{I}_3$, the motion parameters, $\gamma$, $\vec{A}$, and $\vec{V}$, may be computed as follows:

$$\gamma = \|\vec{\Gamma}\|/\Delta T \tag{27}$$
$$\vec{A} = -\vec{\Gamma}/\|\vec{\Gamma}\| \tag{28}$$
$$\vec{V} = \vec{P}/\Delta T - \tfrac{1}{2}\vec{\Gamma} \tag{29}$$

The signs of equations (27) and (28) are chosen to be consistent with equation (18), and so that $\gamma$ is nonnegative.

### 6.7 Pure Translation

For pure translation, $\mathcal{R} = \mathcal{I}_3$, and $\vec{\Gamma} = \vec{0}$. This class of motion has three degrees of freedom. The recursive relationship simplifies to

$$\vec{Q}^s_{i+1,j} = \vec{Q}^s_{i,j} + \vec{V}\Delta T$$

which implies

$$\vec{Q}^s_{i,j} = \vec{Q}^s_{0,j} + t\vec{V} = \vec{Q}^s_{0,j} + i\vec{V}\Delta T$$

The chronogeneous matrix corresponding to this motion is

$$\mathcal{B} = \left[\begin{array}{ccc|c|c} & \mathcal{I}_3 & & \vec{0} & \vec{V}\Delta T \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{30}$$

and the following relationship expresses $\vec{V}$ in terms of the $\vec{P}$ subvector of $\mathcal{B}$:

$$\vec{V} = \vec{P}/\Delta T \tag{31}$$

### 6.8 Structure from Chronogeneous Motion

The author is currently working on solving the structure from motion problem for an object undergoing constant chronogeneous motion. Details of the solutions will be presented in a future paper. Table 1 summarizes the number of frames required to solve this problem for a given number of points, for both rigid and nonrigid motion.

| Rigid Chronogeneous Motion | |
|---|---|
| Points | Frames |
| 1 | 6 |
| 2 | 4 |
| 3 | 3 |

| Nonrigid Chronogeneous Motion | |
|---|---|
| Points | Frames |
| 1 | 9 |
| 2 | 5 |
| 3 | 4 |
| 5 | 3 |

Table 1: Number of Frames Required to Solve SFM Problem

## 7 Recovery of Absolute Depth from a Monocular Image Sequence

In this section, we present a novel application of the methodology developed in this paper. We show how, under certain circumstances, absolute depth may be recovered from a monocular image sequence.

Assume that a (rigid) object, undergoing constant chronogeneous motion, is imaged by a stationary camera (perspective projection). Let us ignore any measurement or correspondence error, and assume that the structure from motion problem has been solved for chronogeneous motion. (We will present a solution to this problem in an upcoming paper). Let

$$\mathcal{B}_T = \left[\begin{array}{ccc|c|c} & \mathcal{R}_T & & \vec{\Gamma}_T & \vec{P}_T \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

be the "true" matrix representing the motion of the object, and let

$$\mathcal{B}_S = \left[\begin{array}{ccc|c|c} & \mathcal{R}_S & & \vec{\Gamma}_S & \vec{P}_S \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

be the computed solution to the structure from motion problem. Then the following relationships hold

$$\mathcal{R}_T = \mathcal{R}_S \tag{32}$$
$$\vec{\Gamma}_T = \lambda\vec{\Gamma}_S \tag{33}$$
$$\vec{P}_T = \lambda\vec{P}_S \tag{34}$$

where $\lambda > 0$ is an unknown scale factor. The above relationships reflect the fact that, without additional assumptions, the depth can only be determined to within an unknown (positive) scale factor.

We now make the additional assumption that the object is accelerating solely due to a constant external force of known magnitude, and show how the scale factor may be recovered. This, in turn, allows the true chronogeneous matrix, and hence the absolute parameters of motion, and the absolute distances to points on the object to be recovered. This assumption is reasonable for certain objects, such as a falling apple, or a cannonball (neglecting air resistance). Such objects are undergoing "ballistic" motion. The object may be rotating, but in order for the motion to be chronogeneous, the direction of the axis of rotation must be aligned with the direction of the external force (gravity in this case). In other words, the axis of rotation must point either "upward" or "downward", with respect to the force vector. In order to determine the scale factor, we use the fact that the *magnitude* of the acceleration is the same in all inertial reference frames. In the following, let $g$ be the acceleration due to the external force.

For the general case of rigid chronogeneous motion, we have from equation (18):

$$\begin{aligned} g &= |\gamma_T| \\ &= |-(\vec{A}_T\cdot\vec{\Gamma}_T)/\Delta T| \\ &= |-(\vec{A}_S\cdot(\lambda\vec{\Gamma}_S))/\Delta T| \\ &= |\lambda(\vec{A}_S\cdot\vec{\Gamma}_S)/\Delta T| \end{aligned}$$

and therefore

$$\lambda = |g\Delta T/(\vec{A}_S\cdot\vec{\Gamma}_S)| = g\Delta T/|\vec{A}_S\cdot\vec{\Gamma}_S| \tag{35}$$

For the subcase of pure acceleration, the above equation holds if we identify the axis of rotation, and the axis of acceleration. However, a more direct derivation is possible. From equation (27):

$$g = |\gamma_T| = \|\vec{\Gamma}_T\|/\Delta T = \|\lambda\vec{\Gamma}_S\|/\Delta T = \lambda\|\vec{\Gamma}_S\|/\Delta T$$

and therefore

$$\lambda = g\Delta T/\|\vec{\Gamma}_S\| \tag{36}$$

## 8 Conclusion and Future Research

In this section, we present what we see as the major contributions of this research. In addition, we discuss related current and future research of the author.

The first contribution of this research is the *general* nature of the representation. A fairly large and interesting class of motion may be represented. Rotation, translation, and fixed axis motion, as well as (possibly restricted forms of) acceleration are all representable. Furthermore, the represention of rigid and nonrigid motion is unified.

Chronogeneous transformation matrices also provide a *compact representation* of a fairly large class of camera/object motion, and allow the *efficient computation* of the motion of computer generated objects. It is straightforward to calculate a chronogeneous matrix given the underlying motion parameters, and vice versa. Chronogeneous coordinates should thus prove very useful in the fields of computer graphics and animation.

Next, this research *unifies the representation of camera and object motion.* The coordinate transformation vector equation provides the connection between the two. Previous researchers have studied problems involving either camera motion, or object motion, but not both simultaneously. Sometimes the distinction between the two has been ignored. This is mainly because so much research has been devoted to the analysis of the two frames case, where camera and object motion are confounded. It is only when at least three frames are available that these two motions can, to a large extent, (locally) be disambiguated.

Finally, this representation *models physically natural motion.* The importance of this fact is that, by taking advantage of the constraints imposed by the spatio-temporal continuity of such motion, we may be able to (and for chronogeneous motion are able to) solve the structure from motion problem using fewer points and/or frames than when only rigidity is imposed. Furthermore, *structure from nonrigid motion* may also be studied.

The author is currently working on solving the structure from motion problem for an object undergoing constant chronogeneous motion. Details of the solutions will be presented in a future paper.

Also, the coordinate transformation vector equation expresses the motion of a point in terms of the camera chronogeneous matrix, and the object chronogeneous matrix. Solutions to this equation will allow the simultaneous recovery of camera and object motion, to within certain inherent ambiguities.

Finally, as a further research problem, it should be possible to estimate the coefficients of the chronogeneous matrix using Kalman filtering techniques. The parameters of motion could then be determined using the equations developed in this paper.

## Acknowledgement

Special thanks go to Dr. Gérard Medioni, who supervised this research. His comments and suggestions resulted in very significant improvement in the organization and understandability of this paper. Any "rough edges" that remain are my own.

## Appendices

## A Useful Formulae Involving Standard Chronogeneous Matrices

In this appendix, we derive formulae for the product of two standard chronogeneous matrices, the inverse of a standard chronogeneous matrix, and the powers of a standard chronogeneous matrix. Standard chronogeneous matrices are closed under each of these operations, and therefore form a group under matrix multiplication.

### A.1 Formulae for Matrix Products

Let

$$C_A = \left[\begin{array}{ccc|c|c} & \mathcal{S}_A & & \vec{\Gamma}_A & \vec{P}_A \\ \hline 0 & 0 & 0 & 1 & \delta t_A \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

and let

$$C_B = \left[\begin{array}{ccc|c|c} & \mathcal{S}_B & & \vec{\Gamma}_B & \vec{P}_B \\ \hline 0 & 0 & 0 & 1 & \delta t_B \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

Then

$$C_A C_B = \left[\begin{array}{ccc|c|c} & \mathcal{S}_A\mathcal{S}_B & & \mathcal{S}_A\vec{\Gamma}_B + \vec{\Gamma}_A & \mathcal{S}_A\vec{P}_B + \vec{\Gamma}_A\delta t_B + \vec{P}_A \\ \hline 0 & 0 & 0 & 1 & \delta t_B + \delta t_A \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

and

$$C_B C_A = \left[\begin{array}{ccc|c|c} & \mathcal{S}_B\mathcal{S}_A & & \mathcal{S}_B\vec{\Gamma}_A + \vec{\Gamma}_B & \mathcal{S}_B\vec{P}_A + \vec{\Gamma}_B\delta t_A + \vec{P}_B \\ \hline 0 & 0 & 0 & 1 & \delta t_A + \delta t_B \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

Standard chronogeneous matrices are, in general, not commutative.

For the special case of the matrix $\mathcal{T}$,

$$\mathcal{T}C = \left[\begin{array}{ccc|c|c} & \mathcal{S} & & \vec{\Gamma} & \vec{P} \\ \hline 0 & 0 & 0 & 1 & \delta t + \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

and

$$C\mathcal{T} = \left[\begin{array}{ccc|c|c} & \mathcal{S} & & \vec{\Gamma} & \vec{\Gamma}\Delta T + \vec{P} \\ \hline 0 & 0 & 0 & 1 & \Delta T + \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

The matrix $\mathcal{T}$ does not, in general, commute with other standard chronogeneous matrices. However, it does commute if $\vec{\Gamma} = \vec{0}$.

### A.2 Formulae for the Inverse of a Matrix

If the deformation submatrix, $\mathcal{S}$, of a standard chronogeneous matrix, $C$, is invertible, then the inverse of $C$ exists and

$$C^{-1} = \left[\begin{array}{ccc|c|c} & \mathcal{S}^{-1} & & -\mathcal{S}^{-1}\vec{\Gamma} & -\mathcal{S}^{-1}(\vec{P} - \vec{\Gamma}\delta t) \\ \hline 0 & 0 & 0 & 1 & -\delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

The above formula may be verified by multiplying the matrix and its inverse. Remember that for a rotation matrix, the inverse of the matrix is simply its transpose. Therefore, when $\mathcal{S}$ is a rotation matrix, call it $\mathcal{R}$, the formula for the inverse becomes:

$$C^{-1} = \left[\begin{array}{ccc|c|c} & \mathcal{R}^T & & -\mathcal{R}^T\vec{\Gamma} & -\mathcal{R}^T(\vec{P} - \vec{\Gamma}\delta t) \\ \hline 0 & 0 & 0 & 1 & -\delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

If $\mathcal{S} = \mathcal{I}_3$, the formula simplifies to:

$$C^{-1} = \left[\begin{array}{ccc|c|c} & \mathcal{I}_3 & & -\vec{\Gamma} & -\vec{P} + \vec{\Gamma}\delta t \\ \hline 0 & 0 & 0 & 1 & -\delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

As a special case, the inverse of the matrix $\mathcal{T}$ is given by

$$\mathcal{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -\Delta T \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

### A.3 Formulae for Integer Powers of a Matrix

The formula for an arbitrary integer power ($\geq 2$) of a standard chronogeneous matrix is given below. This formula is easily proved by induction.

$$C^i = \left[\begin{array}{ccc|c|c} & \mathcal{S}_i & & \vec{\Gamma}_i & \vec{P}_i \\ \hline 0 & 0 & 0 & 1 & i\delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$= \left[\begin{array}{ccc|c|c} & \mathcal{S}^i & & (\sum_{k=0}^{i-1}\mathcal{S}^k)\vec{\Gamma} & (\sum_{k=0}^{i-1}\mathcal{S}^k)\vec{P} + (\sum_{k=0}^{i-2}(i-1-k)\mathcal{S}^k)\vec{\Gamma}\delta t \\ \hline 0 & 0 & 0 & 1 & i\delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

where we use the convention that the $0^{th}$ power of a $(3 \times 3)$ matrix is the $(3 \times 3)$ identity matrix, $\mathcal{I}_3$, and the following recurrence relations hold:

$$\begin{aligned} \mathcal{S}_{i+1} &= (\mathcal{S})\mathcal{S}_i \\ \vec{\Gamma}_{i+1} &= \mathcal{S}\vec{\Gamma}_i + \vec{\Gamma} \\ \vec{P}_{i+1} &= \mathcal{S}\vec{P}_i + \vec{\Gamma}(i\delta t) + \vec{P} \end{aligned}$$

If $\mathcal{S} = \mathcal{I}_3$, then the above formula simplifies to

$$C^i = \left[\begin{array}{ccc|c|c} & \mathcal{I}_3 & & i\vec{\Gamma} & i\vec{P} + \frac{1}{2}i(i-1)\vec{\Gamma}\delta t \\ \hline 0 & 0 & 0 & 1 & i\delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

and holds for all $i \geq -1$. Here we use the convention that the $0^{th}$ power of a $(5 \times 5)$ matrix is the $(5 \times 5)$ identity matrix, $\mathcal{I}_5$. The following analogous formula holds when $i \leq 0$:

$$C^i = \left[\begin{array}{ccc|c|c} & \mathcal{I}_3 & & i\vec{\Gamma} & i\vec{P} - \frac{1}{2}i(i+3)\vec{\Gamma}\delta t \\ \hline 0 & 0 & 0 & 1 & i\delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

The formula for an arbitrary power of the matrix $\mathcal{T}$ is simply:

$$T^i = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & i\Delta T \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

This formula holds for *all* integer powers.

### A.4 Special Matrix Forms

Let $\mathcal{H}$ be the purely spatial and time independent rigid chronogeneous transformation defined as

$$\mathcal{H} = \left[\begin{array}{ccc|c|c} & \mathcal{R} & & \vec{0} & \vec{T} \\ \hline 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

Then

$$\mathcal{H}^{-1} = \left[\begin{array}{ccc|c|c} & \mathcal{R}^T & & \vec{0} & -\mathcal{R}^T\vec{T} \\ \hline 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

and the following relationships hold:

$$\mathcal{HC} = \left[\begin{array}{ccc|c|c} & \mathcal{RS} & & \mathcal{R}\vec{\Gamma} & \mathcal{R}\vec{P} + \vec{T} \\ \hline 0 & 0 & 0 & 1 & \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathcal{H}^{-1}\mathcal{C} = \left[\begin{array}{ccc|c|c} & \mathcal{R}^T\mathcal{S} & & \mathcal{R}^T\vec{\Gamma} & \mathcal{R}^T(\vec{P} - \vec{T}) \\ \hline 0 & 0 & 0 & 1 & \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathcal{HCH}^{-1} = \left[\begin{array}{ccc|c|c} & \mathcal{RSR}^T & & \mathcal{R}\vec{\Gamma} & (\mathcal{I}_3 - \mathcal{RSR}^T)\vec{T} + \mathcal{R}\vec{P} \\ \hline 0 & 0 & 0 & 1 & \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathcal{H}^{-1}\mathcal{CH} = \left[\begin{array}{ccc|c|c} & \mathcal{R}^T\mathcal{SR} & & \mathcal{R}^T\vec{\Gamma} & \mathcal{R}^T(\vec{P} + (\mathcal{S} - \mathcal{I}_3)\vec{T}) \\ \hline 0 & 0 & 0 & 1 & \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

If $\mathcal{S} = \mathcal{I}_3$, then the last two equations may be simplified as follows:

$$\mathcal{HCH}^{-1} = \left[\begin{array}{ccc|c|c} & \mathcal{I}_3 & & \mathcal{R}\vec{\Gamma} & \mathcal{R}\vec{P} \\ \hline 0 & 0 & 0 & 1 & \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathcal{H}^{-1}\mathcal{CH} = \left[\begin{array}{ccc|c|c} & \mathcal{I}_3 & & \mathcal{R}^T\vec{\Gamma} & \mathcal{R}^T\vec{P} \\ \hline 0 & 0 & 0 & 1 & \delta t \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

The following form is useful when $\delta t_{\mathcal{C}} = \Delta T$:

$$\mathcal{TC}^{-1}\mathcal{T} = \left[\begin{array}{ccc|c|c} & \mathcal{S}^{-1} & & -\mathcal{S}^{-1}\vec{\Gamma} & -\mathcal{S}^{-1}\vec{P} \\ \hline 0 & 0 & 0 & 1 & \Delta T \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

## References


[Che85] Su-shing Chen. *Structure-from-motion without the rigidity assumption.* In *Proceedings of the Third Workshop on Computer Vision: Representation and Control*, pages 105–112, IEEE Computer Society Press, October 13–16, 1985.

[CP86] Su-shing Chen and Michael Penna. Recognizing deformations of nonrigid bodies. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 452–455, IEEE Computer Society Press, June 22–26, 1986.

[HP86] David J. Heeger and Alex P. Pentland. Seeing structure through chaos. In *Proceedings of the Workshop on Motion: Representation and Analysis*, pages 131–136, IEEE Computer Society Press, May 7–9, 1986.

[LH74] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems.* Prentice-Hall, Inc., Englewood Cliffs, N. J., 1974. See chapter 7 for basics of pseudoinverse.

[Nag86] Hans-Hellmut Nagel. Image sequences - ten (octal) years - from phenomenology towards a theoretical foundation. In *Proceedings of the Eighth International Conference on Pattern Recognition*, pages 1174–1185, IEEE Computer Society Press, October 27–31, 1986.

[Pau81] Richard P. Paul. *Robot Manipulators: Mathmatics, Programming, and Control*, chapter 1. MIT Press, Cambridge, Massachusetts, 1981. Chapter heading: Homogeneous Transformations.

[Rob68] L. G. Roberts. Machine perception of three-dimensional solids. In J. T. Tippett et al., editors, *Optical and Electro-Optical Information Processing*, pages 159–197, MIT Press, Cambridge, Massachusetts, 1968.

[Sha86] Hormoz Shariat. *The Motion Problem: How to use more than two frames.* PhD thesis, IRIS, School of Engineering, University of Southern California, October 1986.

[Sub86] Muralidhara Subbarao. Interpretation of image motion fields: a spatio-temporal approach. In *Proceedings of the Workshop on Motion: Representation and Analysis*, pages 157–165, IEEE Computer Society Press, May 7–9, 1986.

[Ull79] Shimon Ullman. *The Interpretation of Visual Motion.* MIT Press, Cambridge, Massachusetts, 1979.

# Generic Models for Robot Navigation

David J. Kriegman [†]
Thomas O. Binford
Thilaka Sumanaweera

Robotics Laboratory
Department of Computer Science
Stanford University
Stanford, California 94305

## Abstract

This paper introduces the concept of generic modelling. A particular application is an explicit abstract generic building model which is valid for a broad class of buildings. It is built with mechanisms that have been used to create generic models for other object classes including machine screws and generalized cylinders. These methods extend the generality of object classes which can be implemented, compared to previous implementations of object classes.

Ultimately, given sensor information and a generic model, the model can be instantiated. Using a combination of epipolar stereo vision along with monocular vision, a portion of the generic building model has been instantiated. This instantiation can then be used for mobile robot navigation.

Generic Models will be used to represent object class in the SUCCESSOR vision system.

## 1 Introduction

In much recent work concerning autonomous mobile robotics, there have been two disparate views about the role of geometric modelling of the robot's environment. While some researchers have tried to integrate sensor information into a world model[7,15], others have noted some of the problems, especially brittleness, with forcing the sensor data to fit a previously defined model. They have opted for either a more probabilistic model[8] or a more reactive architecture that does not rely on a world model[5].

We contend that an environmental model will be necessary for autonomously performing intelligent tasks. Furthermore, modelling requires fewer parameters to describe the world which simplifies planning. What exactly do we mean by a model? First, the model should not represent a particular environment. A separate model should not be necessary for every type of building, be it a Frank Loyd Wright house or an I.M. Pei Office building. Instead, one should model features that might be found in any building. We refer to this as a **generic model**. A generic model is a single model that describes a broad class of objects. The problem of modelling class has been looked at by [1,4].

As motivation, consider some of our earlier work. An epipolar stereo vision system was developed which matches vertical edges found crossing the horizon plane; these are projected back into 3-space [9,15]. Using only small vertical segments, a model of the hallway was built. Doors are recognized by the separation of these edges along with complementary grey level changes across opposite sides of a door. A number of coplanar verticals form a wall, and two parallel walls define a hallway. Uncertainty in sensor data was modelled as a multivariate normal distribution and was reduced by Kalman filtering. Given the hallway model, the high level symbolic command "Enter the second door on the left" was executed. Without modelling, the semantics of such a command are not defined.

This system proved to be fast ($< 10$ second cycle times) and fairly robust; however, looking only at the horizon line has obvious disadvantages. Although there is enough information to recognize a door based on only vertical segments, there are other unmodelled objects that fit the door model which are not necessarily doors (e.g. a fuse box). Given tight enough constraints on

*This research was supported in part by a subcontract to Advanced Decision Systems, S1093 S-1 (Phase II). "Knowledge Based Vision Task B," from a contract to the Defense Advanced Research Projects Agency. Partial support was provided by the Air Force Office of Scientific Research under contract F33615-85-C-5106 "Basic Research in Robotics."

†Supported by a fellowship from the Fannie and John Hertz Foundation.

some of the parameters,(i.e. tweaking) the system successfully recognized doors and rejected other objects on the wall. Unfortunately, this implies that the robot may not operate in other buildings with different coloration or different doors sizes.

Of greater importance, in this previous work there was no explicit representation of the building model. Instead, the concept of the generic model was embedded within the lisp code that instantiated the model, making it impossible to reason about the hallway when ambiguities arose. Instead, an explicit model of a generic building is preferred. By automatically reasoning from sensor data and the generic model, portions of the generic model can be instantiated to arrive at a model of a particular building. This paper deals primarily with the representation of a generic model. The generic model is not only applicable to mobile robotics but will be useful for modelling broad classes of objects for general vision research and will be used in the **SUCCESSOR** vision system[2,12].

## 2 Generic Models

Let us consider the generic model of a building, and then explore how a generic model is actually represented. Since a generic model represents a broad class of objects, we cannot draw it since it is underconstrained and ambiguous. However, typical examples can be drawn by choosing typical subclasses and typical structures. Given sensor information, the generic model can be partially constrained until, ultimately, there is an instantiation that is useful for navigation, sensing, planning, and performing tasks. In general, it will not be possible to instantiate the generic model fully, but instead sensing will impose enough constraints necessary for the task. For example, though the height of a building may not be known, height is not needed for navigation between rooms on the same floor. Furthermore, the generic model should be related to realistic physical processes that actually constrain the modelled class. For example, the fact that buildings are used by people yields constraints on the size of doors and other parts of the building.

Beyond parameterizing the model, the generic model allows for gross changes in object geometry and topology. This permits different numbers of components as well as different component types.

Since the generic model constrains surfaces found in a scene, it will constrain the location of discontinuities in the surfaces. These are generically observable, and so there are constraints on the observables in images which can be used for instantiating the model.[2]

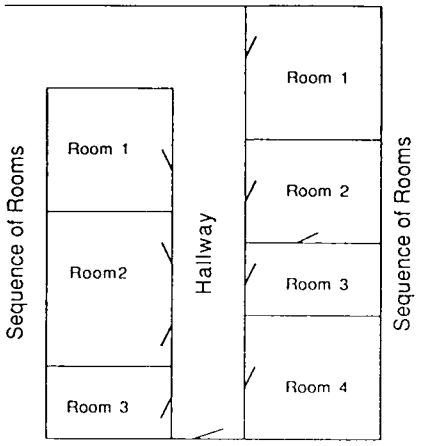



**Figure 1.** An instance of the generic building.

### 2.1 A Generic Building Model

Consider the typical section of a building shown in figure 1. A generic building should be described in terms of its function or purpose, as well as physical constraints. Since the primary purpose of a building is environmental isolation, a building is closed. One of the primary physical processes constraining a building is gravity; a vector represents the direction of gravity. A floor, which is planar for motion efficiency, is generally normal to the gravity vector.

Because floors are constrained to be normal to the gravity vector, buildings are divided up into multiple stories. On each floor, the need for further isolation leads to multiple rooms separated by walls. Walls are often planar, with the gravity vector lying in the plane of the wall. More generally, walls are ruled surfaces, and the rulings are parallel to the gravity vector. Because of stability and gravity, there must be contact between the wall and the floor. This constrains the number of degrees of freedom (DOF) of the position of a planar wall from six to three.

Though the purpose of a building is environmental isolation, there must be a means for movement between rooms within the building, and so there are movable coverings or doors. Because of gravity, the potential energy of the door should not change between the open and closed state of the door. This further constrains the door to move in a plane, limiting its motion to three DOF. More typically of course, doors are hinged but sliding doors are also common.

Furthermore, the primary occupants of a building are people, and so the scale of a building is determined accordingly. The minimum dimensions of such building elements as doors, ceilings, and walls are constrained

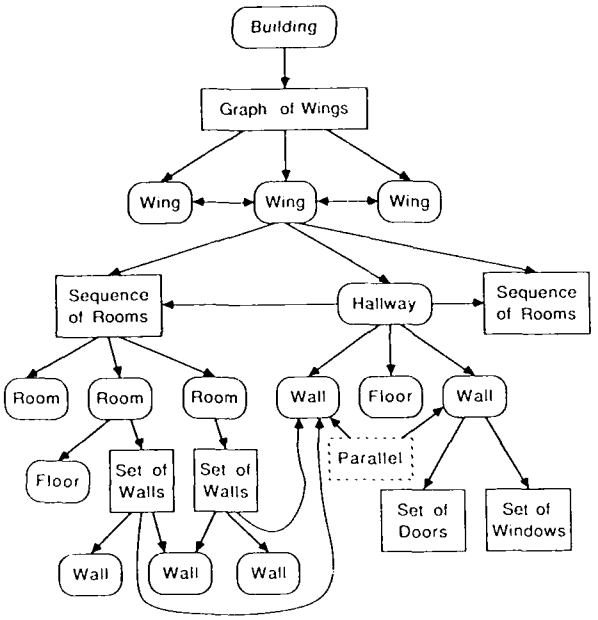



**Figure 2.** Part of the structure of the generic building model.

by the size of a typical large person. This put a greatest lower bound on these dimensions. Economics places an upper bound; it is often too expensive to make something larger than necessary, and so the upper bound may be related to the lower bound.

A generic building is divided up into stories. Each has a floor and is divided into rooms connected by doors. The need for isolation implies that a special room is necessary for movement between rooms without disturbing the occupants of other rooms; this is a hallway. purpose is a channel for motion between other rooms. This purpose constrains the hallway to be wide enough for two people to pass, so a hall must be wider than twice the width of a large person. Because, locally, buildings are organized around hallways, multiple wings are also found. Figure 2 shows the representation while figure 1 shows an instance of the model.

Further constraints may be less obvious. Doors do not overlap each other along a wall. Rooms are must be taller than people. Since a hallway is a type of room, it must also be taller than a person.

Of course. if the primary purpose of the building is not human isolation, then the dimensions must be related to that purpose. A door in the final assembly building at Cape Kennedy is related to the size of a Saturn V rocket, not the size of a person.

Thus, the generic model should be related to the processes that constrain it. Object dimension is usually not arbitrary but related to either these processes or object function. Note that the generic model has not said very much about the overall shape of the building. It must allow for changes in building topology, the number and interconnectivity of rooms. It must allow for different types of rooms, different types of walls and doors.

The generic model does not represent a typical building, but instead it represents all typical buildings. Unfortunately, for every constraint mentioned, there are exceptions. Our aim is that the generic building model will cover a broad class of buildings. This seems to be achieved. We believe that extensions to buildings not describable by this generic model, like the Guggenheim Museum and igloos, can be made compactly.

## 2.2 The structure of a Generic Model

Now, consider the representation of a generic model which for the present is primarily concerned with shape. A model is composed of the following five aspects: **classes, sets, numbers, mappings** and **constraints**.

The generic model of an object is a named class (e.g. screws) and is made up of named components and constraints. Components are typed, and the types may be either another class (e.g. a number, a mapping, a plane, a door, etc), a set, or an element of a set. Thus, a hallway might be represented as having components, among others, length, height, a floor of type plane, and two walls each of which is of type wall. The walls may be composed of a plane and a set of doors and windows.

A set may have elements which are either classes or themselves sets. Sets need not be finitely enumerated but may be infinite sets where membership is determined by the set theoretic definition of membership, satisfaction of a constraint (predicate). Set operations on infinite sets are represented as boolean operations on the constraints of the set (e.g. the intersection of two sets is the conjunction of the constraints). Thus, all set operations (e.g. union, intersection, difference, etc.) are defined for both finite and infinite sets. Clearly, an infinite set cannot be enumerated.

Additionally, classes can represent mappings from a domain to a range. This is used to represent geometric objects according to their mathematical definitions. For example, curves are represented as a mapping from an open set in R1 to a higher space. Surfaces are a set of mappings from R2 into R3. Given these general mappings or possibly relations, the mathematical definition of object geometry can be defined.

Constraints describe the relationship between components and subcomponents. They may simply be algebraic and boolean constraints between components

with a numeric type such as the pythagorean theorem for the lengths of the legs and hypotenuse of a right triangle. They may constrain the element types or cardinality of sets. Constraints may also be symbolic, such as two planar faces being parallel. Symbolic constraints are defined by name along with the type of the objects that are are being constrained. This allows for geometric reasoning [16] without having to do more costly symbolic algebraic reasoning. Symbolic constraints may be expandable into algebraic, boolean or even further geometric constraints. By expanding symbolic constraints into algebraic constraints, algebraic constraint manipulation can be used.[4] Furthermore, constraints may be quantified over sets. For example a constraint on the doors of a wall might be: $\forall x \in doors(wall) : height(x) < height(wall)$.

Since classes are named, one must be careful about their scope; A class is always named with respect to a particular namespace. Classes themselves may also have local namespaces containing, perhaps, the component types used in that class. Namespaces are arranged in a tree and classes within the parent are inherited by the children modulo shadowing. For example, the class of screws may be contained in a global root namespace. The screw class contains a local namespace which may contain a class called head. Thus, if a component of a screw is of type head, this will not be confused with biological heads. Furthermore, the screw may still reference classes named in the global namespace (e.g. real numbers, cylinders).

Finally, classes are defined in an object-oriented manner; a class may be defined as a subclass of another class or classes leading to taxonomies describable by a directed acyclic graph (DAG). A subclass is a strict specialization of the parent classes. Any constraint that is true for a parent class, is true for a child. A subclass will inherit components and constraints of the parent classes, leading to issues of multiple inheritance[1]. An example is presented below when discussing the representation of generic model of generalized cylinders and shown in figure 3. This requirement of specialization is a different requirement for multiple inheritance than in traditional object oriented systems such as flavors and CLOS. These systems rely on the partial ordering of ancestors in a superclass tree rather than of specialization.

Since subclasses are specializations, a subclass inherits all of the components of all of the parent classes. If a component is defined by multiple ancestors with differing types, then the type is determined by iteratively comparing ancestor types. If the multiply inherited component type is a class, then the more specialized type is used as determined by the specialization DAG of all classes. If neither class is a specialization of the other, then an attempt is made to create automatically a new class which is a specialization of the types of the component from the two ancestor classes. This is not always possible because certain types may be incompatible.

If the multiply inherited components have set types, then the specialization of the type is the intersection of the two type sets. For example, if there is a class of machine screws which may have cylindrical, round fillister, bolt and flat heads, and a class of screws using allen wrench sockets with heads that are either cylindrical or round; head type is represented as an element of a set of heads. The subclass of screws of these two parent class would have either cylindrical or round head types.

The constraints on the subclasses of a class are the union of all of the constraints of the parent. This is similar to ACRONYM's restriction graph[4].

Generic models have been used to represent generalized cylinders (GC)[3]. By using specialization to resolve multiple inheritance of components, a taxonomy of generalized cylinders similar to Shafer's[13] has been built and is shown in fig. 3. The components of a GC are a spine, cross section, and sweeping rule. As an example of subclass inheritance the spine is represented as a curve which is a map from R1 to R3. A straight homogeneous generalized cylinder (SHGC)[12,13], which is a specialization of a GC, has a straight spine. Since a straight line is a specialization of a general curve, any class that is a subclass of an SHGC and GC will have a spine which is a straight line.

Another aspect to note about the representation of generalized cylinders is that the spine and cross section are explicitly represented as a curve and a surface, which as mentioned earlier are represented as sets of mappings. The sweeping rule can also be represented as a mapping. So the actual mathematical definition of "What is a generalized cylinder?" is explicit in the generic model. Furthermore, since "primitives" such as generalized cylinders are actually defined in the generic modelling system, higher level generic models are not confined to use any particular type of primitive such as GC's or polyhedra.

## 3 Some experimental results

To date, the representation in the previous section has been implemented and applied to modelling three classes of objects: screws and bolts, hallways, and generalized cylinders. The last class allows us to model SHGC's and acts as an interface to the modelling system described in [12].

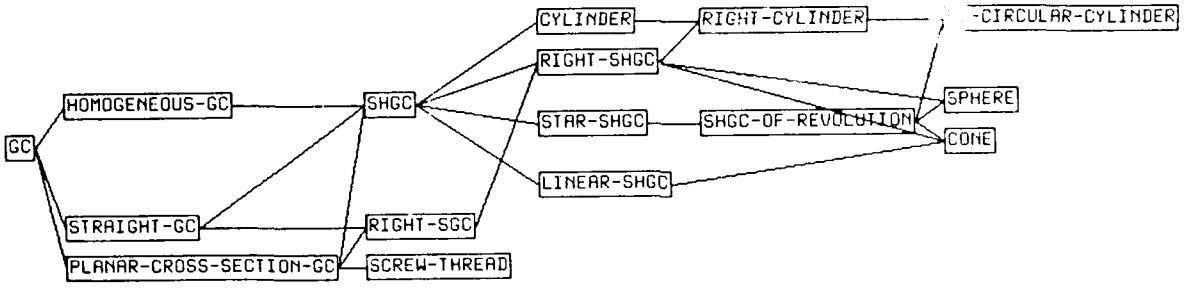



**Figure 3**. A taxonomy of Generalized Cylinders from the Generic Modelling System.

### 3.1 Generic Screw Model

Machine Screws are a common object in almost every assembly task. They are also typical of man made objects in that they can be easily classified; a taxonomy of screws can be created.

First, screws come in many siz , with different lengths and diameters. They can be divided into two components, a head and a shaft. The head may have many different shapes (e.g. round head, flat head, bolt head, etc). Also, the head may have a slot which may be a phillips slot, a straight slot, an allen slot, etc. Of course the size of these head features scale with the screw size. Screw shafts also vary in their shape. Machine screws typically have a cylindrical shaft while sheet metal screws taper to a point. The tip of a screw may be blunt, pointed or chamfered. Finally, there is variability in the threads. The threads can be of different pitch and even different cross sections (i.e. wood and acme threads).

Except for the threads, all of these components lend themselves to representations of straight homogeneous generalized cylinder primitives; the SHGC primitives were described earlier. Threads can be represented as a generalized cylinder with a helical spine with an appropriate planar cross section. Figure 4 shows some of the components of the generic screw model. Screw size is simply parameterized by a pair of positive real numbers representing length and radius. The head, slot and tip are elements of a set of possibilities. Additionally, there are components describing the thread. Finally, constraints describe the relationship of the various components. For example, the size of the head is related to the size of the shaft. The shaft and head are coaxial. The "head-assembly" is created by taking the set difference of the head and the slot.

A taxonomy of screws can be created by further constraining the generic model. For example, the class of flat head phillips machine screws are created by constraining the head to be a flat head instead of any element of the set of heads and the slot is constrained accordingly. Furthermore, the size of the screws may be constrained to be an element of a set of actual screw sizes (e.g. two subclasses for metric and english screws).

Figure 5 shows some instances of the generic screw that were created by instantiating components with values that satisfy all of the constraints. The instance of the generic model is translated into a representation understandable by the modeling system of [12] which is used for performing set operations and rendering. At present we only have an approximate method for rendering the threads and have not yet integrated even this.

### 3.2 Instantiating the Generic Hallway

The hallway model provides a starting point for developing a generic building model. The generic hallway is modeled as two side walls (cuboids), a floor (a plane), and for each wall, a set of doorways and doors. Constraints relate the direction of walls and floor to the gravity vector. Sizes of these objects are related to typical human dimensions.

We have instantiated part of this generic model using a combination of stereo and monocular vision. At this time, the strategy for instantiation has been hand programmed into lisp code. However, given this strategy, the model is automatically instantiated from images. In the future the strategies will be automatically derived from the generic building model and the generic observability model. Starting with the stereo pair in figure 6-a, the 3D locations of vertical lines crossing the horizon plane are determined[15] as shown in 6-b. We attempt to instantiate a door based on pairs of verticals edges. To confirm further the partially instantiated door, edges in the monocular image corresponding to the sides of the door found by a modified version of Canny's edge detector [6] have been linked and tracked[14] to determine whether they contact the

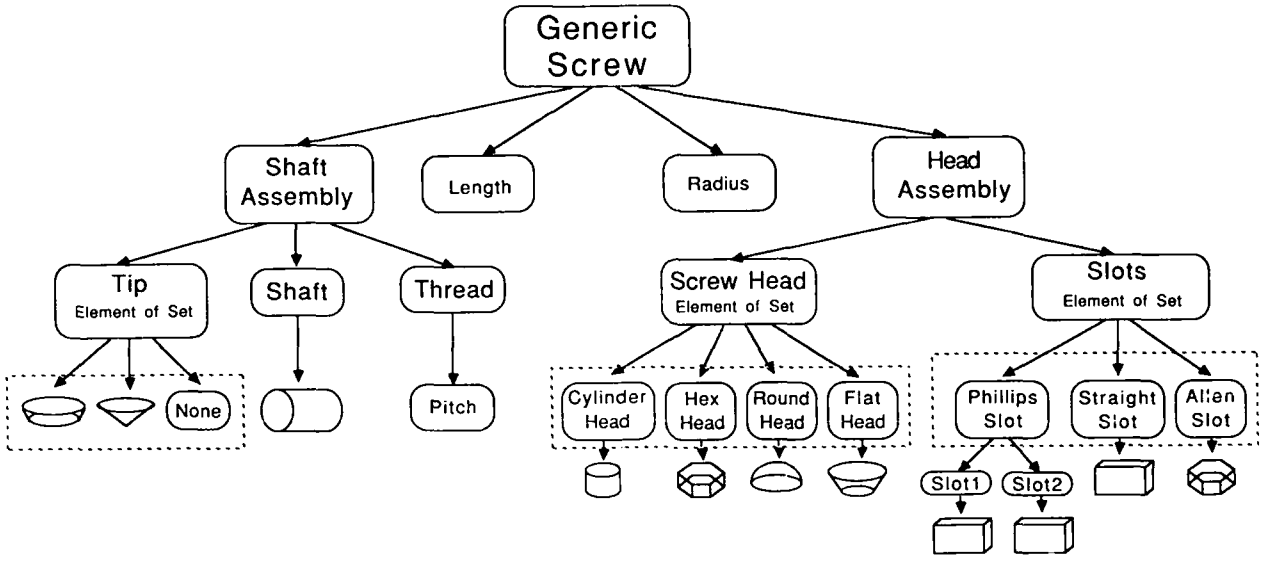



**Figure 4.** Some of the components of a generic screw.

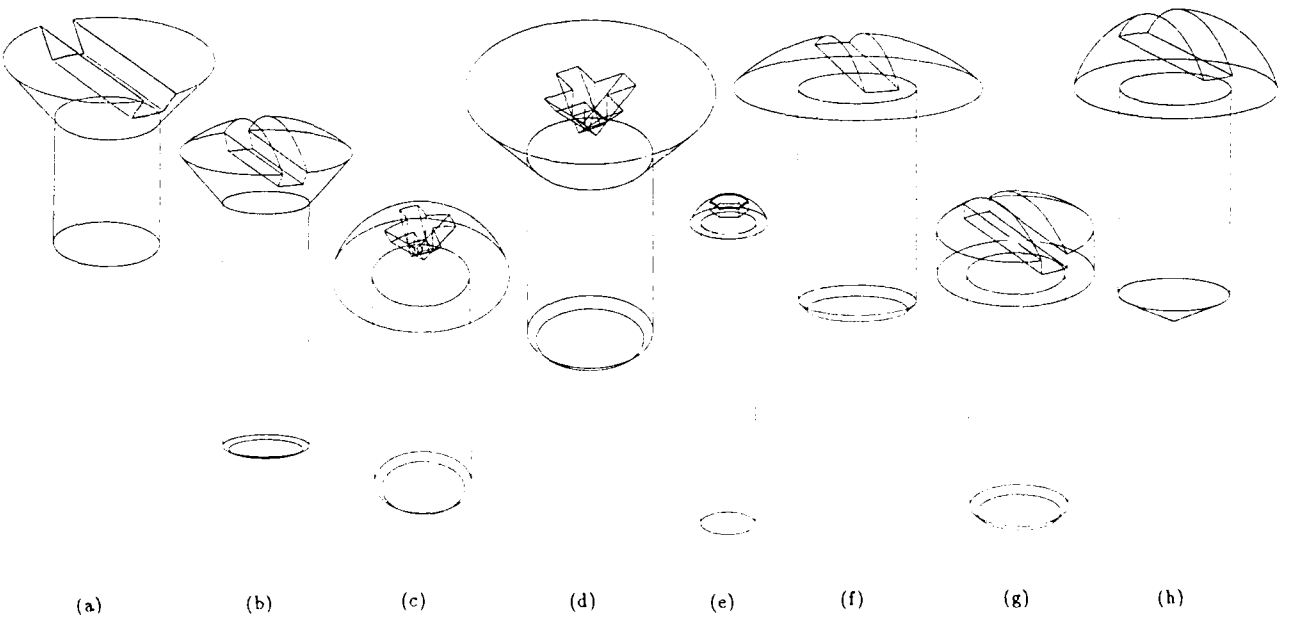



**Figure 5.** A pot-pouri of instances of the generic screw: (a) Flat head, (b) Oval head, (c) Round phillips head, (d) Flat phillips head, (e) Round head with hex slot, (f) Truss head, (g) Fillister head, (h) Round head

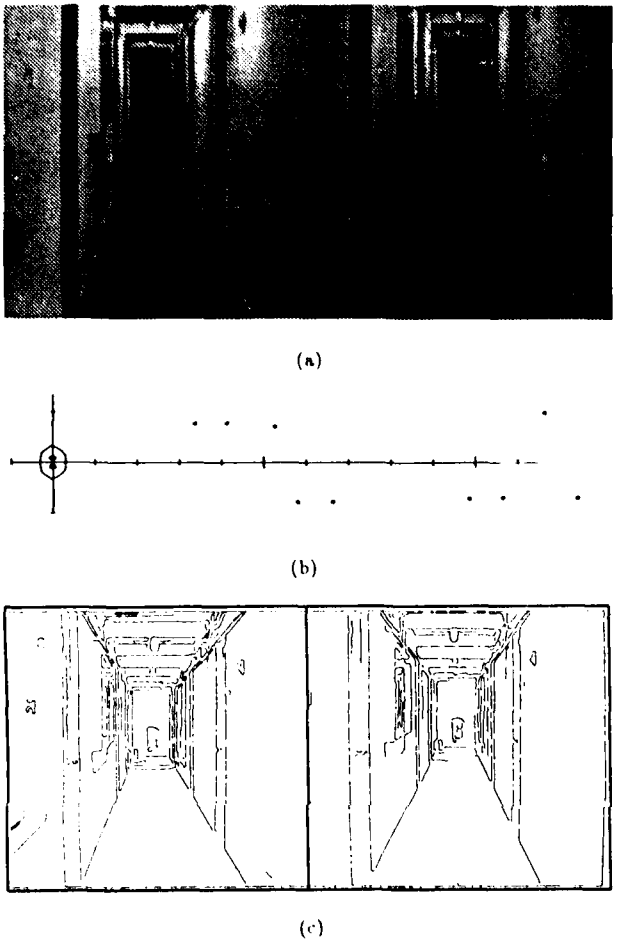


**Figure 6.** (a) A stereo view of the hallway, (b) The projection of the stereo correspondences into the ground plane, (c) The hallway edges.

floor, a requirement from the generic model. See figure 6-c. Given the 3D location of an edgel from the epipolar stereo pair and the length of the vertical from that edgel to the floor in the image, the length in 3 space can be determined. Finally, verticals are tracked to the top of the door to determine the door's height. The height of the door can then be instantiated, again checking whether it satisfies such constraints as being larger than a human.

Now, given a set of doors found in an image, they can be grouped into sets of coplanar doors. Doorway coplanarity is a constraint on walls; this is used to instantiate a wall. Two parallel walls form the side walls of a hallway. There are still undetermined parameters of the hallway, (e.g. length and height). However, from the constraints and the already instantiated values, bounds on these other parameters can be found. For example, the height of a wall is known to be greater than the height of all of the doors from the generic model. Figure 7 shows an instance of the hallway model built from the image in figure 6 where bounds on the uninstantiated components were found and then instantiated with random values within the bounds. The height of the doors was found to be within 5 percent of the true heights The primary source of error is in trying to determine the location of the end points of the verticals in the images. There was a bit more variability in the door widths. The far door is noticeably narrower than the nearer doors; the stereo uncertainty model accounts for this[9]. This instantiated model can then be used for navigation.

## 4 Generic Models and Navigation

Ultimately, the generic building model will be used as a basis for planning motion of the mobile robot. The generic model helps produce a hierarchical framework for simplifying motion planning. The generic model indicates that motion planning can first be done at the level of a graph of rooms and and their interconnection, doorways. Furthermore, without instantiating a model, the generic model indicates that the main purpose of hallways is isolating rooms and allowing motion *between rooms. Thus, it is probable that the motion* will be along hallways, first leaving one room, via a door, traveling down a series of hallways and then entering another room. Thus, at this level, the planner is only interested in which doorways have to be crossed. This is typically the level at which a person might give directions to another person.

At a lower level, a traditional motion planner could determine paths within a room. However, these planners generally have a fairly high computational cost. Additionally, the layout of the entire room is not always known at planning time, and there is always uncertainty in sensor information, so the expensive motion planner may have to be invoked very often. A more interesting approach will be to use a local motion planner with guaranteed convergence [10]. From our instantiated generic model, artificial boundaries can be created so that the local planner will not create paths that cross doorways into undesired rooms. Since typical rooms have a reasonable number of obstacles covering a finite area, this method will converge in a reasonable amount of time.

At present, these ideas of motion planning have not been implemented. Our efforts have concentrated on developing the generic model and attempting to instantiate it. Once a generic model is instantiated, the previously mentioned local methods and graph searching

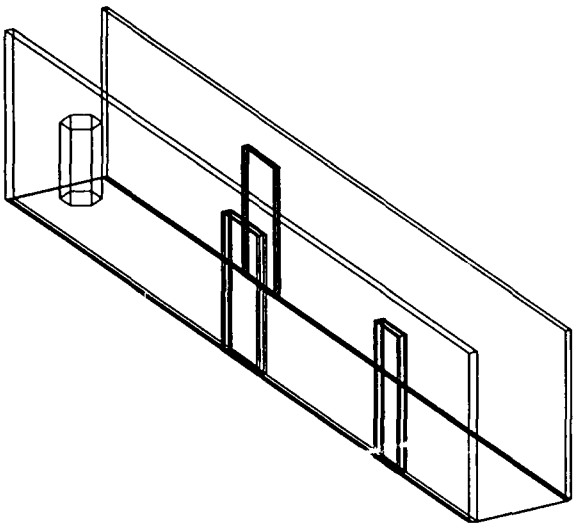


**Figure 7.** An instance of the hallway generic model built from sensor information. The hexagonal object represents the robot.

should be straight forward. While the graph planner decides which doors and rooms to traverse, the local planner will find paths between doors within a room. Because of the high reliance on doors as critical navigation landmarks, it is important to recognize them with high confidence. The local planner will certainly not produce an optimal path; however, this does not seem important because an autonomous mobile robot will seldom repeat the same path. Also, locally rooms are often changing. Thus, locally, paths through rooms will be changing (e.g. chairs, waste paper baskets, children toys and people are always in motion). The same situation will seldom be encountered. Thus, little is gained by putting effort into finding the "optimal" shortest time path. Instead, effort should be spent determining good paths through the static graph of rooms.

**Acknowledgements**

We would like to thank Ernst Triendl and Jean Ponce for their contributions to this work.

## References

[1] L. Barford. *A Graphical Language-Based Editor for Generic Solid Models Represented by Constraints.* PhD thesis, Cornell Stanford University Computer Science Dept., 1987.

[2] T. O. Binford. Generic surface interpretation: observability model. In *International Symposium on Robotics Research*, 1987.

[3] T. O. Binford. Visual perception by computer. In *IEE Conf. on Systems and Control*, 1971.

[4] R. Brooks. *Symbolic Reasoning among 3-D Models and 2-D Images.* PhD thesis, Stanford University Computer Science Dept., 1981.

[5] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 14–23, March 1986.

[6] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Machine Intell.*, 679–98, Nov. 1986.

[7] J. Crowley. Navigation for an intelligent mobile robot. *IEEE Journal of Robotics and Automation*, 31–41, March 1985.

[8] A. Elfes and L. Matthies. Sensor integration for robot navigation: combing sonar and stereo range data in a grid-based representation. In *IEEE Conference on Decision and Control*, 1987.

[9] D. Kriegman, E. Triendl, and T. Binford. A mobile robot: sensing, planning and locomotion. In *IEEE International Conference on Robotics and Automation*, 1987.

[10] V. Lumelsky. Alogorithmic issues of sensor-based robot motion planning. In *IEEE Conference on Decision and Control*, 1987.

[11] J. Ponce, D. Chelberg, D. Kriegman, and W. Mann. Geometric modelling with generalized cylinders. In *IEEE Workshop on Computer Vision*, 1987.

[12] S. Shafer and T. Kanade. *The Theory of Straight Homogonous Generalized Cylinders and a Taxonomy of Generalized Cylinders.* Technical Report CMU-CS-83-105, Carnegie-Mellon University, 1983.

[13] T. Sumanaweera, J. Ponce, G. Healey, and B. Lee. Integrated segmentation using geometrical and physical constraints. In *Image Understanding Workshop*, 1988.

[14] E. Triendl and D. Kriegman. Stereo vision and navigation within buildings. In *IEEE International Conference on Robotics and Automation*, 1987.

[15] J. M. Wing and F. Arbab. *Geometric Reasoning: A new paradign for processing geometric information.* Technical Report CMU-CS-85-144, Carnegie-Mellon University, 1985.

# Mathematical Morphology and the Morphological Sampling Theorem

*Robert M. Haralick*

Intelligent Systems Laboratory
Department of Electrical Engineering
University of Washington
Seattle, WA 98195

*ABSTRACT*

For the purpose of object or defect identification required in vision applications in manufacturing, the operations of mathematical morphology are perhaps more useful than the convolution operations. This is because mathematical morphology provides a natural algebraic theory for working with shape. The morphological sampling theorem described in this paper states how a digital image must be morphologically filtered before sampling in order to preserve relevant information after sampling. The sampling theorem indicates to what precision an appropriately morphologically filtered image can be reconstructed after sampling, and it specifies the relationship between morphological processing before sampling and the more computationally efficient scheme of morphologically operating on the sampled image. Thus, the morphological sampling theorem provides a sound basis for recognizing and extracting shape information in a computationally efficient, multi-resolution pyramid processing approach.

## 1. Overview

Section 2 briefly describes the vision group at the University of Washington. Sections 3 and 4 review the basic morphological operations and the algebra defined by them. Sections 5 through 7 develop the binary and grayscale morphological sampling theorem. Section 8 briefly discusses other vision research being done at the University of Washington and where we expect to be going.

## 2. UW Vision Group

The University of Washington has assembled a vision group whose senior researchers are internationally known and recognized for their contributions in image analysis and computer vision. The primary members of the group are Robert M. Haralick, Linda G. Shapiro, Steven L. Tanimoto, Kenneth Sloan, John Palmer, Arun Somani and Mani Soma. Professor Robert Haralick holds the Boeing Clairmont Egtvedt Chair in Electrical Engineering and has an adjunct appointment in Computer Science. He has published several hundred papers on a variety of vision topics including texture, facet model, consistent labeling, and most recently on mathematical morphology. His texture papers showed the utility of the gray tone co-occurrence matrix. The facet model papers showed that a variety of low-level feature extraction operations such as edge, line, and corner detection can be viewed as possessing a locally estimated underlying gray tone intensity surface of which the given image is a sampled noisy version. The consistent labeling papers recognized the variety of vision problems which are special cases of consistent labeling. These papers developed the general theory behind the relaxation and look ahead techniques which speed up the tree search.

Professor Haralick has served as head of the IEEE Computer Society Pattern Analysis and Machine Intelligence Technical Commiittee. He now serves as the head of the Computer Society Task Force on Artificial Intelligence. He is a Fellow of the IEEE for his contributions in image processing and computer vision. He serves on the Editorial Board of the *IEEE Transactions on Pattern Analysis and Machine Intelligence*. He is the computer vision area editor for *Communications of the ACM*. He also serves as an associate editor for *Computer Vision, Graphics, and Image Processing* as well as *Pattern Recognition*.

Professor Linda G. Shapiro holds an appointment in the Electrical Engineering Department in the Computer Engineering program, as well as an adjunct appointment in Computer Science. She is the Editor of *Computer Vision, Graphics, and Image Processing*, the journal which is the grandfather journal for archival quality papers on computer vision. Professor Shapiro has designed a language and associated recognition system for expressing the structural relationshps among entities in an image. She has defined a structural representation for describing two-dimensional shapes and implemented an associated shape matching procedure. She has developed an inexact matching methodology which permits, in special cases, efficient comparison (polynomial time) of two structures which are not identical as well a method for organizing relational models into clusters of similar models so that an unknown object can be compared only against the cluster representatives. She has

worked on the design of INSIGHT, a dataflow language for expressing vision and AI algorithms in parallel architectures.

Professor Steven L. Tanimoto holds an appointment in the Computer Science Department (which has been acknowledged to be one of the top ten in the country) and an adjunct appointment in the Electrical Engineering Department. He is the Editor in Chief of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* which has historically established itself as the premiere journal for vision research. Professor Tanimoto works in the area of parallel architecture and algorithms for machine vision. He has formulated a hierarchical architecture for high-speed vision. This architecture embodies several novel features. One of these is the use of a parallel-pyramidal interconnection network allowing both local and global image transforms to be computed rapidly. Another is hardware for a parallel operator for hierarchical cellular logic that can transform every cell in parallel according to all the values in its 14-point hierarchical neighborhood, in a single machine instruction. On the algorithms side, Professor Tanimoto has developed a fast Hough transform algorithm for the architecture; it uses a novel bottom-up clustering procedure that achieves $O(\log N)$ computation times. He has also developed several fast image search algorithms. At the meta-algorithm level, he has formulated a model for automatic algorithm development in the machine vision context; the model uses concepts from state-space as well as image processing.

Dr. Kenneth Sloan holds an appointment as Assistant Professor in the Computer Science Department. He has experience in computer graphics, vision, artificial intelligence, and networks. His dissertation at the University of Pennsylvania treated the problem of analyzing monocular, color views of natural outdoor scenes. At the University of Rochester, he worked on a broad range of vision problems as part of the DARPA Image Understanding program. He worked with Peter Selfridge on the use of adaptive low-level vision in the domain of aerial image understanding. While at the Architecture Machine Group at the Massachusetts Institute of Technology, he worked primarily in the domain of 3D reconstruction and display of (usually biological) surfaces determined from series of 2D slices. He has been at the University of Washington for over 3 years, during which time he has directed the establishment of the GRaphics and Artificial Intelligence Lab (GRAIL). His recent work on reconstruction and display of the human retina was featured on the cover of *Science*.

Dr. John Palmer holds an appointment as Assistant Professor in the Psychology Department. He has been conducting research in human vision and human performance. He has investigated the features mediating vision for spatial relations and motion. The research defined a general mathematical model of a feature and generated predictions that can be used to distinguish between alternative hypotheses of what features mediate a particular perceptual judgment. In experimental work, he tested alternative feature theories for localizing moving stimuli. The experiments demonstrated that motion contributes a unique feature to visual localization under certain temporal conditions and not under others. These experiments provided a rigorous confirmationof informal introspections concerning perceived motion. The feature theory and experimental methodology developed in these studies can be readily generalized to other perceptual domains.

Dr. Palmer has also investigated various limitations on human performance. Several recent studies have measured the limits on human perception imposed by attention and memory. These studies have provided new quantifications of the capacity limits of human observers. In previous studies, he has searched for good predictors of skilled performance in reading and has studied what learning conditions allow for highly-skilled performance in laboratory categorization tasks.

Dr. Arun Somani holds an appointment as an Assistant Professor in Electrical Engineering. He has several years of experience in developing real time systems and does research in computer architecture, neural networks, parallel processing and fault tolerant computing. In his dissertation at McGill University he developed a parallel fault diagnosis algorithm for a multi-processing environment and a theory characterizing partial diagnosable systems. Earlier, he designed a VLSI architecture for performing various searches and maintaining parallel data dictionaries. He recently developed a parallel pipelined architecture for determining line of sight visibility for graphics application. His current research interest is in special purpose parallel computer architectures for high-performance and high reliability systems.

Dr. Mani Soma holds an appointment as an Assistant Professor in Electrical Engineering after receiving his Ph.D. degree from Stanford University in 1980 and working for two years at the General Electric Research and Development Center (Schenectady, New York). His research interests include the design, testing and reliability characterization of integrated circuits and systems. Since 1984 he has focused on the architectures and designs of application-specific integrated circuits (ASICs), where the emphsis is on high-performance digital-analog systems for applications in vision, image processing, and signal processing. Testability and reliability aspects of these circuits are emphasized and considered in performance tradeoffs.

## 3. Overview and Summary

Mathematical morphology provides an approach to the processing of digital images which is based on shape. Appropriately used, mathematical morphological operations

tend to simplify image data, preserving their essential shape characteristics and eliminating irrelevancies. As the identification of objects, object features, and assembly defects correlate directly with shape, it becomes apparent that the natural processing approach to deal with the manufacturing machine vision recognition process and the visually guided robot problem is mathematical morphology.

Machines which perform mophologic operations are not new. They are the essence of what cellular logic machines such as the Golay logic processor (Golay, 1969), Diff3 (Graham and Norgren, 1980), PICAP (Kruse, 1977), the Leitz Texture Analysis System TAS (Klein and Serra, 1977), the CLIP processor arrays (Duff, 1979), and the Delft Image Processor DIP (Gerritsen and Aardema, 1981) all do. A number of companies now manufacture industrial vision machines which incorporate video rate morphological operations. These companies include International Robomation Inc., Allen Bradley, 3M, Machine Vision International, Maitre, Synthetic Vision Systems, Vicom, Applied Intelligence Systems, Inc., and Leitz. Most of the real time VME and multibus machine vision boards developed by companies such as Datacube, Recognition Technology Inc., and Imaging Technology Inc. all support morphological operations.

The 1985 IEEE Computer Society Workshop on Computer Architecture For Pattern Analysis and Image Database Management had an entire session devoted to computer architecture specialized to perform morphological operations. Papers included those by McCubbrey and Lougheed (1985), Wilson (1985), Kimmel, Jaffe, Manderville, and Lavin (1985), Leonard (1985), Pratt (1985), and Haralick (1985). Gerritsen and Verbeek (1984) show how convolution followed by a table look up operation can accomplish binary morphologic operations.

Although the techniques are being used in the industrial world, the basis and theory of mathematical morphology tend to be (with the exception of the highly mathematical books by Matheron (1975) and Serra (1982) and the more readable chapter in the book by Dougherty and Giardina (1987)) not covered in the textbooks and, until recently, not covered in the journals which discuss image processing or computer vision.

Not only are the operations of mathematical morphology natural for shape processing, but morphological operations on images have relevance to the entire suite of conditioning, labeling, grouping, extracting and matching image processing operations. Thus from low-level to intermediate to high-level vision, morphological techniques are important. Indeed, many successful machine vision algorithms employed in industry on the factory floor, processing thousands of images per day in each application, are based on morphological techniques. Among the recent research papers on morphology are Crimmons and Brown (1985), Zhuang and Haralick (1985), Lee, Haralick, and Shapiro (1987), Haralick, Sternberg, and Zhuang (1987), and Maragos and Schafer (1987). The September 1986 issue of *Computer Vision, Graphics, and Image Processing* is devoted to morphology.

Many well-known relationships worked out in the classical context of the convolution operation have morphological analogs. In this paper, we introduce the digital morphological sampling theorem, which relates to morphology as the standard sampling theorem relates to signal processing and communications. The sampling theorem permits the development of a precise multiresolution approach to morphological processing.

Multiresolution techniques (Ahuja and Samy, 1984; Klinger, 1984; Meresereau and Speake, Tanimoto, 1982; Uhr, 1983) have been useful for at least two fundamental reasons: (1) the representation they provide naturally permits a computational mechanism to focus on objects or features likely to be at least a given specified size (Crowley, 1984; Miller and Stout; Rosenfeld, 1983; Witkin, 1984), and (2) the computational mechanism can operate on only those resolution levels which just suffice for the detection and localization of objects or features of specified size while significantly reducing the number of operations performed (Burt, 1984; Dyer, 1982; Lougheed and McCubbrey, 1980).

The usual resolution hierarchy, called a pyramid, is produced by low pass filtering and then sampling to generate the next lower resolution level of the hierarchy. The basis for a morphological pyramid requires a morphological sampling theorem which explains how an appropriately morphologically filtered and sampled image relates to the unsampled image. It must explain what kinds of shapes are preserved and what kinds are suppressed or eliminated. It must explain the relationship between performing a less costly morphological filtering operation on the sampled image and performing the more costly equivalent morphological filtering operation on the original image. It is just these issues which we address in this paper.

The following results are shown to be true under reasonable morphological sampling conditions. Before sets are sampled, they must be morphologically simplified by an opening or a closing. Such sampled sets can be reconstructed in two ways, by either a closing or a dilation. In both reconstructions, the sampled reconstructed sets are equal to the sampled sets. For binary morphology a set contains its reconstruction by closing and is contained in its reconstruction by dilation; indeed, these are extremal bounding sets. That is, the largest set which downsamples to a given set is its reconstruction by dilation; the smallest is its reconstruction by closing. Furthermore, the distance from the maximal reconstruction to the minimal reconstruction is no more than the diameter of the reconstruction structuring element. Equivalent relations hold in the grayscale morphology. Morphological sampling thus provides reconstructions positioned only to within some spa-

tial tolerance which depends on the sampling interval. This spatial limitation contrasts with the sampling reconstruction process in signal processing from which only those frequencies below the Nyquist frequency can be reconstructed.

A number of relationships follow from the morphological sampling theorem. These relationships govern the commutivity between sampling and then performing morphological operations in the sampled domain versus first performing the morphological operations and then sampling. We find that sampling a minimal reconstruction which has been dilated is identical to dilating the sample set with a sampled structuring element. Sampling a maximal reconstruction which has been eroded is identical to eroding the sampled set with a sampled structuring element. These results establish bounds which can be used to determine the difference between morphological operations in the sampled domains and operations in the original domain followed by sampling.

All set morphological relationships are immediately generalizable to gray scale morphology via the umbra homomorphism theorems. For grayscale images, the bounds which the reconstruction establishes are bounds which are simultaneously grayscale and spatial.

## 4. The Morphological Operations

The language of mathematical morphology is that of set theory. Sets in mathematical morphology represent the shapes which are manifested on binary or gray tone images. The set of all the black pixels in a black and white image, (a binary image) constitutes a complete description of the binary image. Sets in Euclidean 2-space denote foreground regions in binary images. Sets in Euclidean 3-space may denote time varying binary imagery or static grayscale imagery as well as binary solids. Sets in higher dimensional spaces may incorporate additional image information, like color, or multiple perspective imagery. Mathematical morphological transformations apply to sets of any dimensions, those like Euclidean N-space, or those like its discrete or digitized equivalent, the set of N-tuples of integers, $Z^N$. For the sake of simplicity we will refer to either of these sets as $E^N$.

Those points in a set being morphologically transformed are considered as the selected set of points and those in the complement set are considered as not selected. Hence, morphology from this point of view is binary morphology. We begin our discussion with the binary morphological operations of dilation and erosion and then extend this discussion to gray scale morphology..

### 4.1 Dilation and Erosion

Dilation is the morphological transformation which combines two sets using vector addition of set elements. If $A$ and $B$ are sets in N-space ($E^N$) with elements $a$ and $b$ respectively, $a = (a_1, ..., a_N)$ and $b = (b_1, ..., b_N)$ being $N$-tuples of element coordinates, then the dilation of $A$ by $B$ is the set of all possible vector sums of pairs of elements, one coming from $A$ and one coming from $B$.

Let $A$ and $B$ be subsets of $E^N$. The dilation of $A$ by $B$ is denoted by $A \oplus B$ and is defined by

$$A \oplus B = \{c \in E^N | c = a + b \text{ for some } a \in A \text{ and } b \in B\}$$

Erosion is the morphological dual to dilation. It is the morphological transformation which combines two sets using the vector subtraction of set elements. If $A$ and $B$ are sets in Euclidean N-space, then erosion of $A$ by $B$ is the set of all elements $x$ for which $x + b \in A$ for every $b \in B$. Some image processing people use the name *shrink* or *reduce* for erosion.

The erosion of $A$ by $B$ is denoted by $A \ominus B$ and is defined by

$$A \ominus B = \{x \in E^N | x + b \in A \text{ for every } b \in B\}.$$

For any set $A \subseteq E^N$ and $x \in E^N$, let $A_x$ denote the *translation* of $A$ by $x$;

$$A_x = \{y \mid \text{for some } a \in A,\ y = a + x\}.$$

For any set $A \subseteq E^N$, let $\check{A}$ denote the *reflection* of $A$ about the origin;

$$\check{A} = \{x \mid \text{ for some } a \in A,\ x = -a\}.$$

Relationships satisfied by dilation and erosion include the following:

$$A \oplus B = B \oplus A$$
$$(A \oplus B) \oplus C = A \oplus (B \oplus C)$$
$$(A \ominus B) \ominus C = A \ominus (B \oplus C)$$
$$(A \cup B) \oplus C = (A \oplus C) \cup (B \oplus C)$$
$$(A \cap B) \ominus C = (A \ominus C) \cap (B \ominus C)$$
$$A \oplus B = \bigcup_{b \in B} A_b$$
$$A \ominus B = \bigcup_{b \in B} A_{-b}$$
$$A \subseteq B \Rightarrow A \oplus C \subseteq B \oplus C$$
$$A \subseteq B \Rightarrow A \ominus C \subseteq B \ominus C$$
$$(A \cap B) \oplus C \subseteq (A \oplus C) \cap (B \oplus C)$$
$$(A \cup B) \ominus C \supseteq (A \ominus C) \cup (B \ominus C)$$
$$(A \oplus B)^C = A^C \ominus \check{B}$$
$$A \ominus (B \cup C) = (A \ominus B) \cap (A \ominus C)$$

### 4.2 Opening and Closing

In practice, dilations and erosions are usually employed in pairs, either dilation of an image followed by the erosion of the dilated result, or image erosion followed by dilation. In either case, the result of iteratively applied dilations and erosions is an elimination of specific image detail smaller than the structuring element without the global geometric distortion of unsuppressed features. For example, opening

an image with a disk structuring element smooths the contour, breaks narrow isthmuses, and eliminates small islands and sharp peaks or capes. Closing an image with a disk structuring element smooths the contours, fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps on the contours.

Of particular significance is the fact that image transformations employing iteratively applied dilations and erosions are idempotent, that is, their reapplication effects no further changes to the previously transformed result. The practical importance of idempotent transformations is that they comprise complete and closed stages of image analysis algorithms because shapes can be naturally described in terms of under what structuring elements they can be opened or can be closed and yet remain the same. Their functionality corresponds closely to the specification of a signal by its bandwidth. Morphologically filtering an image by an opening or closing operation corresponds to the ideal non-realizable bandpass filters of conventional linear filtering. Once an image is ideal bandpassed filtered, further ideal bandpass filtering does not alter the result.

These properties motivate the importance of opening and closing, concepts first studied by Matheron (1967, 1975) who was interested in axiomatizing the concept of size. Both Matheron's (1975) definitions and Serra's (1982) definitions for opening and closing are identical to the ones given here, but their formulas appear different because they use the symbol $\ominus$ to mean Minkowski subtraction rather than erosion.

The morphological filtering operations of opening and closing are made up of dilation and erosion performed in different orders. The *opening* of $A$ by $B$ is defined by

$$A \circ B = (A \ominus B) \oplus B.$$

The *closing* of $A$ by $B$ is defined by

$$A \bullet B = (A \oplus B) \ominus B.$$

Opening and closing satisfy the following basic relationships:

$$(A \circ B) \circ B = A \circ B$$
$$(A \bullet B) \bullet B = A \bullet B$$
$$A \circ B \subseteq A$$
$$A \subseteq A \bullet B$$
$$A \subseteq B \Rightarrow A \circ C \subseteq B \circ C$$
$$A \subseteq B \Rightarrow A \bullet C \subseteq B \bullet C$$
$$(A \circ B)^C = A^C \bullet \check{B}$$
$$(A \bullet B)^C = A^C \circ \check{B}$$
$$A \circ B = (A \circ B) \circ B$$
$$(A \circ B) = (A \circ B) \bullet B$$
$$A \oplus B = (A \bullet B) \oplus B$$
$$A \oplus B = (A \oplus B) \circ K$$

The reason that openings and closings deal directly with shape properties is apparent from the following representation theorem for openings.

$$A \circ B = \{x \mid \text{for some } y,\ x \in B_y \subseteq A\}.$$

$A$ opened by $B$ contains only those points of $A$ which can be covered by some translation $B_y$ which is, in turn, entirely contained inside $A$. Thus $x$ is a member of the opening if it lies in some area inside $A$ which entirely contains a translated copy of the shape $B$. In this sense, $A$ opened by $B$ is the set of all points of $A$ which can participate in areas of $A$ which match $B$. If $B$ is a disk of diameter $d$, for example, then $A \circ B$ would be that part of $A$ which in no place is narrower than $d$.

The duality relationship $(A \circ B)^C = A^C \bullet \check{B}$ between opening and closing implies a corresponding representation theorem for closing

$$A \bullet B = \{x \mid x \in \check{B}_y \text{ implies } \check{B}_y \cap A \neq \emptyset\}.$$

$A$ closed by $B$ consists of all those points $x$ for which $x$ being covered by some translation $\check{B}_y$ implies that $\check{B}_y$ "hits" or intersects some part of $A$. A more extensive discussion of these relationships can be found in Haralick, Sternberg, and Zhuang (1987).

### 4.3 Gray Scale Morphology

The binary morphological operations of dilation, erosion, opening and closing are all naturally extended to gray scale imagery. The extensions, due to Sternberg (1980, 1982b), keep all the relationships previously discussed in a form suitable for grayscale image data. Peleg and Rosenfeld (1981) use grayscale morphology to generalize the medial axis transform to gray scale imaging. Peleg, Naor, Hartley, and Avnir (1984) use gray scale morphology to measure changes in texture properties as a function of resolution. Werman and Peleg (1985) use gray scale morphology for texture feature extraction. Favre, Muggli, Stucki, and Bonderet (1985) use gray scale morphology for the detection of platelet thrombosis detection in cross sections of blood vessels. Coleman and Sampson (1985) use gray scale morphology on range data imagery to help mate a robot gripper to an object. Maragos and Schafer (1987) discuss the relationship between median filtering and morphology.

We will develop the extension in the following way. First we introduce the concept of the top surface of a set and the related concept of the umbra of a surface. Then gray scale dilation will be defined as the surface of the dilation of the umbras. From this definition we will proceed to the representation which indicates that gray scale dilation can be computed in terms of a maximum operation on a set of sums. A similar plan is followed for erosion which can be evaluated in terms of a minimum operation on a set of differences.

of each other. Then we illustrate how the umbra operation is a homomorphism from the gray scale morphology to the binary morphology. Having the homomorphism in hand, all the interesting relationships follow by appropriately unwrapping and wrapping the involved sets or functions.

### 4.3.1 Grayscale Dilation and Erosion

We begin with the concepts of surface of a set and the umbra of a surface. Suppose a set $A$ in Euclidean N-space is given. We adopt the convention that the first (N-1) coordinates of the N-tuples of $A$ constitute the spatial domain of $A$ and the $N^{th}$ coordinate is for the surface. For grayscale imagery, $N = 3$. The top or top surface of $A$ is a function defined on the projection of $A$ onto its first (N-1) coordinates. For each (N-1)-tuple $x$ the top surface of $A$ at $x$ is the highest value $y$ such that the $N$–tuple $(x, y) \in A$. This is illustrated in Figure 1. If the space we work in is Eucidean, we can express this using the concept of supremum. If the space is discrete, we use the more familiar concept of maximum. Since we have supressed the underlying space in what follows, we use maximum throughout. The careful reader will want to translate maximum to supremum under the appropriate circumstances.

Let $A \subseteq E^N$ and $F = \{x \in E^{N-1} \mid \text{for some } y \in E,\ (x, y) \in A\}$. The top or top surface of $A$, denoted by $T[A] : F \rightarrow E$, is defined by

$$T[A](x) = \max \{y \mid (x, y) \in A\}$$

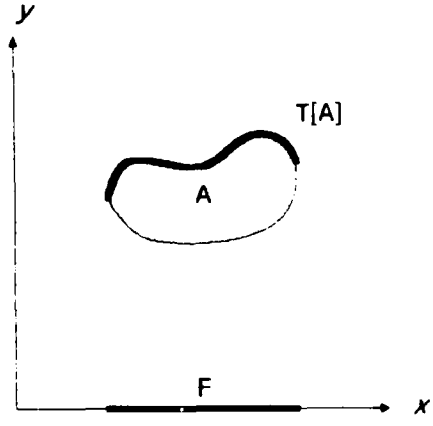



**Figure 1** *illustrates the concept of top or top surface of a set.*

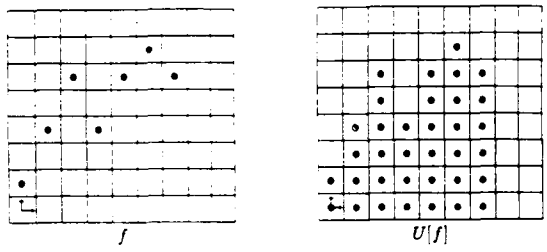
A set $A \subseteq E^{N-1} \times E$ is an umbra if and only if $(x, y) \in A$ implies that $(x, z) \in A$ for every $z \leq y$.

For any function $f$ defined on some subset $F$ of Euclidean $(N-1)$-space the umbra of $f$ is a set consisting of the surface $f$ and everything below the surface. Let $F \subseteq E^{N-1}$ and $f : F \rightarrow E$. The umbra of $f$, denoted by $U[f]$, $U[f] \subseteq F \times E$, is defined by

$$U[f] = \{(x, y) \in F \times E \mid y \leq f(x)\}$$

Obviously, the umbra of $f$ is an umbra.

Example This illustrates a discretized one dimensional function $f$ defined on a domain consisting of seven successive column positions and a finite portion of its umbra which lies on or below the function $f$. The actual umbra has infinite extent below $f$.

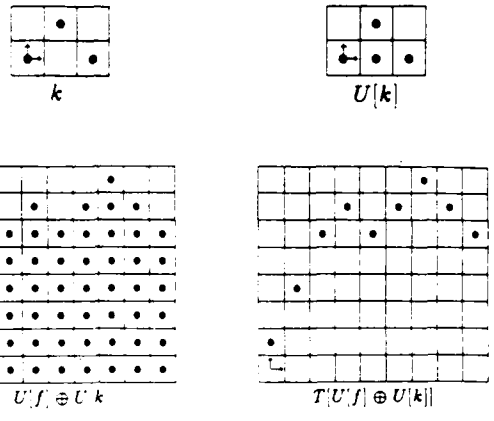



Having defined the operations of taking a top surface of a set and the umbra of a surface, we can define grayscale dilation. The gray scale dilation of two functions is defined as the surface of the dilation of their umbras.

Let $F, K \subseteq E^{N-1}$ and $f : F \rightarrow E$ and $k : K \rightarrow E$. The dilation of $f$ by $k$ is denoted by $f \oplus k$, $f \oplus k : F \oplus K \rightarrow E$, and is defined by

$$f \oplus k = T[U[f] \oplus U[k]]$$

Example This illustrates a second discretized one-dimensional function $k$ defined on a domain consisting of three successive column positions and a finite portion of its umbra which lies on or below the function $k$. The dilation of the umbras of $f$ (from the previous example) and $k$ are shown and the surface of the dilation of the umbras of $f$ and $k$ are shown.



The definition of grayscale dilation tells us conceptually how to compute the gray scale dilation, but this conceptual way is not a reasonable way to compute it in hardware. The following proposition establishes that grayscale dilation can be accomplished by taking the maximum of a set of sums. Hence, grayscale dilation has the same complexity as convolution. However, instead of doing the summation of products as in convolution, a maximum of sums is performed.

Proposition Let $f : F \to E$ and $k : K \to E$. Then $f \oplus k : F \oplus K \to E$ can be computed by

$$(f \oplus k)(x) = \max_{\substack{z \in K \\ x-z \in F}} \{f(x-z) + k(z)\}$$

Proof Suppose $z = (f \oplus k)(x)$. Then $z = T[U[f] \oplus U[k]](x)$. By definition of surface,

$$z = \max\{y \mid (x,\ y) \in [U[f] \oplus U[k]]\}.$$

By definition of dilation,

$$z = \max\{a+b \mid \text{for some } u \in K \text{ satisfying } x-u \in F,\ (x-u,\ a) \in U[f] \text{ and } (u,\ b) \in U[k]\}$$

By definition of umbra, the largest $a$ such that $(x-u,\ a) \in U[f]$ is $a = f(x-u)$. Likewise, the largest $b$ such that $(u,\ b) \in U[k]$ is $b = k(u)$. Hence

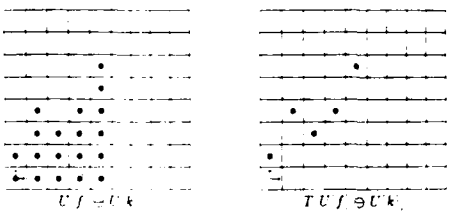
$$\begin{aligned} z &= \max\{f(x-u) + k(u) \mid u \in K,\ (x-u) \in F\} \\ &= \max_{\substack{u \in K \\ (x-u) \in F}} \{f(x-u) + k(u)\} \end{aligned}$$

The definition for grayscale erosion proceeds in a similar way to the definition of grayscale dilation. The grayscale erosion of one function by another is the surface of the binary erosions of the umbra of one with the umbra of the other.

Let $F \subseteq E^{N-1}$ and $K \subseteq E^{N-1}$. Let $f : F \to E$ and $k : K \to E$. The erosion of $f$ by $k$ is denoted by $f \ominus k$, $f \ominus k : F \ominus K \to E$, and is defined by

$$f \ominus k = T[U[f] \ominus U[k]].$$

Example Using the same function $f$ and $k$ of the previous example, illustrated here is the erosion of $f$ by $k$ by taking the surface of the erosion of the umbra of $f$ by the umbra of $k$.




Evaluating a grayscale erosion is accomplished by taking the minimum of a set of differences. Hence its complexity is the same as dilation. Its form is like correlation with the summation of correlation replaced by the minimum operation and the product of correlation replaced by a subtraction operation. If the underlying space is Euclidean, substitute infimum for minimum.

Proposition Let $f : F \to E$ and $k : K \to E$. Then $f \ominus k : F \ominus K \to E$ can be computed by $(f \ominus k)(x) = \min_{z \in K}\{f(x+z) - k(z)\}$

Proof Suppose $z = (f \ominus k)(x)$. Then, $z = T[U[f] \ominus U[k]](x)$. By definition of surface, $z = \max\{y \mid (x,\ y) \in U[f] \ominus U[k]\}$. By definition of erosion

$$z = \max\{y \mid \text{for every } (u,\ v) \in U[k], \text{ there exists } x \text{ such that } (x,\ y) + (u,\ v) \in U[f]\}$$

By definition of umbra,

$$\begin{aligned} z &= \max\{y \mid \text{for every } u \in K,\ v \le k(u),\ y + v \le f(x+u)\} \\ &= \max\{y \mid \text{for every } u \in K,\ v \le k(u),\ y \le f(x+u) - v\} \end{aligned}$$

But $y \le f(x+u) - v$ for every $v \le k(u)$ implies $y \le f(x+u) - k(u)$. Hence,

$$z = \max\{y \mid \text{for every } u \in K,\ y \le f(x+u) - k(u)\}$$

But $y \le f(x+u) - k(u)$ for every $u \in K$ implies

$$y \le \min_{u \in K}[f(x+u) - k(u)].$$

Now,

$$\begin{aligned} z &= \max\{y \mid y \le \min_{u \in K}[f(x+u) - k(u)]\} \\ &= \min_{u \in K}[f(x+u) - k(u)] \end{aligned}$$

The basic relationship between the surface and umbra operations is that they are, in a certain sense, inverses of each other. More precisely, the surface operation will always undo the umbra operation. That is, the surface operation is a left inverse to the umbra operation. However the umbra operation is not an inverse to the surface operation. Without any constraints on the set $A$, the strongest statement which can be made is that the umbra of the surface of $A$ contains $A$. When the set $A$ is an umbra, then the umbra of the surface of $A$ is itself $A$. In this case the umbra operation is an inverse to the surface operation.

Having established that the surface operation is always an inverse to the umbra operation and that the umbra operation is the inverse to the surface operation when the set being operated on itself is an umbra, we only need to note that the dilation of one umbra by another is an umbra and that the erosion of one umbra by another is also an umbra and we are ready to develop the umbra homomorphism theorem.

The umbra homomorphism theorem states that the operation of taking an umbra is a homomophism from the

gray scale morphology to the binary morphology.

Umbra Homomorphism Theorem: Let $F, K \subseteq E^{N-1}$ and $f : F \to E$ and $k : K \to E$. Then

$$(1)\ U[f \oplus k] = U[f] \oplus U[k] \quad \text{and}$$
$$(2)\ U[f \ominus k] = U[f] \ominus U[k]$$

Proof (1) $f \oplus k = T[U[f] \oplus U[k]]$ so that $U[f \oplus k] = U[T[U[f] \oplus U[k]]]$. But $U[f] \oplus U[k]$ is an umbra and for sets which are umbras the umbra operation undoes the surface operation. Hence $U[f \oplus k] = U[T[U[f] \oplus U[k]]] = U[f] \oplus U[k]$.

(2) $f \ominus k = T[U[f] \ominus U[k]]$ so that $U[f \ominus k] = U[T[U[f] \ominus U[k]]]$. But $U[f] \ominus U[k]$ is an umbra and for sets which are umbras, the umbra operation undoes the surface operation. Hence,

$$U[f \ominus k] = U[T[U[f] \ominus U[k]]] = U[f] \ominus U[k]$$

To illustrate how the umbra homomorphism property is used to prove relationships by first wrapping the relationship by re-expressing it in terms of umbra and surface operations and then transforming it through the umbra homomorphism property and finally by unwrapping it using the definitions of grayscale dilation and erosion, we state and prove the commutivity and associativity of grayscale dilation and the chain rule for grayscale erosion.

Proposition $f \oplus k = k \oplus f$

Proof

$$\begin{aligned} f \oplus k &= T[U[f] \oplus U[k]] \\ &= T[U[k] \oplus U[f]] \\ &= k \oplus f \end{aligned}$$

Proposition $k_1 \oplus (k_2 \oplus k_3) = (k_1 \oplus k_2) \oplus k_3$

Proof

$$\begin{aligned} k_1 \oplus (k_2 \oplus k_3) &= T[U[k_1] \oplus U[k_2 \oplus k_3]] \\ &= T[U[k_1] \oplus (U[k_2] \oplus U[k_3])] \\ &= T[(U[k_1] \oplus U[k_2]) \oplus U[k_3]] \\ &= T[U[k_1 \oplus k_2] \oplus U[k_3]] \\ &= (k_1 \oplus k_2) \oplus k_3 \end{aligned}$$

Proposition $(f \ominus k_1) \ominus k_2 = f \ominus (k_1 \oplus k_2)$

Proof

$$\begin{aligned} (f \ominus k_1) \ominus k_2 &= T[U[f \ominus k_1] \ominus U[k_2]] \\ &= T[(U[f] \ominus U[k_1]) \ominus U[k_2]] \\ &= T[U[f] \ominus (U[k_1] \oplus U[k_2])] \\ &= T[U[f] \ominus U[k_1 \oplus k_2]] \\ &= f \ominus (k_1 \oplus k_2) \end{aligned}$$

Relationships satisfied by gray dilation and erosion include

$$\begin{gathered} f \oplus g = g \oplus f \\ (f \oplus g) \oplus h = f \oplus (g \oplus h) \\ (f \ominus g) \ominus h = f \ominus (g \oplus h) \\ \max\{f, g\} \oplus h = \max\{f \oplus h, g \oplus h\} \\ \min\{f, g\} \ominus h = \min\{f \ominus h, g \ominus h\} \\ f < g \Rightarrow f \oplus h < g \oplus h \\ f < g \Rightarrow f \ominus h < f \ominus g \\ \min\{f, g\} \oplus h \leq \min\{f \oplus h, g \oplus h\} \\ \max\{f, g\} \ominus h > \max\{f \ominus h, g \ominus h\} \\ -(f \oplus g) = (-f) \ominus \check{g} \\ f \ominus \max\{g, h\} = \min\{f \ominus g, f \ominus h\} \end{gathered}$$

The first three properties for grayscale dilation are algebraically similar to three properties of convolution.

$$\begin{gathered} f * g = g * f \\ (f * g) * h = f * (g * h) \\ (f + g) * h = (f * h) + (g * h) \end{gathered}$$

This similarity strongly suggests the richness of the underlying algebraic structure for the grayscale morphological operations, despite the fact that they are highly non-linear.

Grayscale opening and closing are defined in an anologous way to opening and closing in the binary mophology and they have similar properties. The grayscale opening of $f$ by structuring element $k$ is denoted by $f \circ k$ and is defined by $f \circ k = (f \ominus k) \oplus k$. The grayscale closing of $f$ by structuring element $k$ is denoted by $f \bullet k$ and is defined by $f \bullet k = (f \oplus k) \ominus k$.

There is a geometric interpretation to the grayscale opening and to the grayscale closing in the same manner that there is a geometric meaning to the binary morphological opening and closing. To obtain the opening of $f$ by a paraboloid structuring element, for example, take the paraboloid, apex up, and slide it under all the surface of $f$ pushing it hard up against the surface. The apex of the paraboloid may not be able to touch all points of $f$. For example, if $f$ has a spike narrower than the paraboloid, the top of the apex may only reach as far as the mouth of the spike. The opening is the surface of the highest points reached by some part of the paraboloid as it slides under all the surface of $f$.

We have not mentioned the duality relationship between grayscale dilation and erosion. We need this in order to give the geometric interpretation to closing. The gray scale duality relationship is analogous to the binary duality relationship. Before stating it, we need the definition of gray scale reflection.

Definition Let $f : F \to E$. The reflection of $f$ is denoted by $\check{f}$, $\check{f}: \check{F} \to E$, and is defined by $\check{f}(x) = f(-x)$.

Grayscale Dilation Erosion Duality Theorem:

Let $f : F \to E$ and $k : K \to E$. Let $x \in (F \oplus K) \cap (F \ominus \check{K})$ be given. Then $-(f \oplus k)(x) = ((-f) \ominus \check{k})(x)$.

It follows immediately from the gray scale dilation and erosion duality that there is a grayscale opening and closing duality.

Grayscale Opening and Closing Duality Theorem:

$$-(f \circ k) = (-f) \bullet \check{k}$$

Having the grayscale opening and closing duality, we immediately have $f \bullet k = -((-f) \circ \check{k})$. In essence, this means that we can think of closing like opening. To close $f$ with a paraboloid structuring element, we take the reflection of the paraboloid, turn it upside down (apex down), and slide it all over the top of the surface of $f$. The closing is the surface of all the lowest points reached by the sliding paraboloid.

Relationships satisfied by gray scale opening and closing include the following:

$$(f \circ g) \circ g = f \circ g$$
$$(f \bullet g) \bullet g = f \bullet g$$
$$f \circ g \leq f$$
$$f \leq f \bullet g$$
$$f \leq g \Rightarrow f \circ h \leq g \circ h$$
$$f \leq g \Rightarrow f \bullet h \leq g \bullet h$$
$$-(f \circ g) = (-f) \bullet \check{g}$$
$$-(f \bullet g) = (-f) \circ \check{g}$$
$$\check{} \ominus g = (f \circ g) \ominus g$$
$$f \ominus g = (f \ominus g) \bullet g$$
$$f \oplus g = (f \bullet g) \oplus g$$
$$f \oplus g = (f \oplus g) \circ g$$

## 5. Morphological Sampling Theorem

The preliminary part of this section sets the stage, discussing the appropriate morphological simplifying and filtering to be done before sampling. Certain relationships must be satisfied between the sampling set and the structuring element used for reconstruction. The main body of the section discusses two kinds of reconstructions of the sampled images: a maximal reconstruction accomplished by dilation and a minimal reconstruction accomplished by closing. Fundamental set bounding relationships are stated which indicate that the closing reconstruction of a set must be contained in the set itself which, in turn, must be contained in its dilation reconstruction. The closing reconstruction differs from the dilation reconstruction by just a dilation by the reconstruction structuring element, so the set bound relationships translate to geometric distance relationships. The section concludes by defining a suitable set distance function which measures the distance between the sampled set and the morphologically filtered set. The distance between the minimal reconstruction and the maximal reconstruction, and the distance between the morphologically filtered set and either of its reconstructions, are all less than the sampling distance.

The first conceptual issue which arises in developing a morphological sampling theorem is how to remove small objects, object protrusions, object intrusions and holes before sampling. It is exactly the presence of this kind of small detail before sampling which causes the sampled result to be unrepresentative of the original, just as in signal processing, the presence of frequencies higher than the Nyquist frequency causes the sampled signal to be unrepresentative of the original signal. This "aliasing" means that signals must be low pass filtered before sampling. Likewise in morphology, the sets must be morphologically filtered and simplified before sampling. Small objects and object protrusions can be eliminated by a suitable opening operation. Small object intrusions and holes can be eliminated by a suitable closing. Since opening and closing are duals, we develop our motivation by just considering the opening operation.

Opening a set $F$ by a structuring element $K$ in order to eliminate small details of $F$ raises, in turn, the issue of how $K$ should relate to the sampling set $S$. If the sample points of $S$ are too finely spaced, little will be accomplished by the reduction in resolution. On the other hand, if $S$ is too coarse relative to $K$, objects preserved in the opening may be missed by the sampling. $S$ and $K$ can be coordinated by demanding that there be a way to reconstruct the opened image from the sampled opened image. Of course, details smaller than $K$, are removed by the opening and cannot be reconstructed.

One natural way to reconstruct a sampled opening is by dilation. If $S$ and $K$ were coordinated to make the reconstructed image (first opened, then sampled, and then dilated) the same as the opened image, we would have a morphological sampling theorem nearly identical to the standard sampling theorem of signal processing. However, morphology cannot provide a perfect reconstruction, as is illustrated by the following one-dimensional continuous domain example.

Let the image $F$ be the union of three topologically open intervals

$$F = (3.1,\ 7.4) \cup (11.5,\ 11.6) \cup (18.9,\ 19.8),$$

where $(x, y)$ denotes the topologically open interval between $x$ and $y$. We can remove all details of less than length 2 by opening with the structuring element $K = (-1,\ 1)$

consisting of the toplogically open interval from -1 to 1. Then the opened image $F \circ K = (3.1, 7.4)$. What should the corresponding sample set be? Consider a sampling set $S = \{x \mid x \text{ an integer}\}$, with a sample spacing of unity; other spacings such as .2, .5 or .7 could illustrate the same sampling concept as well. The sampled opened image $(F \circ K) \cap S = \{4,5,6,7\}$. Dilating by $K$ to reconstruct the image produces $[(F \circ K) \cap S] \oplus K = (3,8)$, an interval which properly contains $F \circ K$. The dilation fills in between the sample points, but cannot "know" to expand on the left end by a length of .9 and yet expand by .4 on the right end. However, the reconstruction is the largest one for which the sampled reconstruction $\{[(F \circ K) \cap S] \oplus K\} \cap S$ produces the sampled opening $(F \circ K) \cap S = \{4,5,6,7\}$. This is easily seen in the example because substituting the closed interval $[3,8]$ for the open interval $(3,8)$ produces the sampled closed interval $[3,8] \cap S = \{3,4,5,6,7,8\}$ which properly contains $(F \circ K) \cap S = \{4,5,6,7\}$.

*The difficulty in reconstructing a sampled opened image* morphologically can be understood in terms of the standard sampling theorem. Consider the case of a precise constant binary valued image. The required morphological simplification means that details smaller than $K$ have been removed from all objects on the opened image, but this removal does not bandlimit the image. In fact the opened image belongs to a special class of infinite bandwidth signals, wherein reconstructing the sampled opened image as specified by *the standard sampling theorem cannot produce the kind* of aliasing found in Moire patterns. The standard sampling *theorem reconstruction produces* a bandlimited signal which passes through the sample points. Thus, the step-like patterns, like the open intervals of $F$, get reconstructed with ringing throughout and with overshoot and undershoot at step edges. By contrast, the morphological reconstruction cannot produce ringing, but the position of any step edge is uncertain within the sampling interval.

In the remainder of this section we give a complete derivation of the results illustrated in the example. First, note that to use a structuring element $K$ as a "reconstruction kernel," $K$ must be large enough to ensure that the dilation of the sampling set $S$ by $K$ covers the entire space $E^N$. For technical reasons apparent in the derivations, we also require that $K$ be symmetric, $K = \check{K}$. In the standard sampling theorem, the period of the highest frequency present must be sampled at least twice in order to properly reconstruct the signal from its sampled form. In mathematical morphology, there is an analogous requirement. The sample spacing must be small enough that the diameter of K is just smaller than these two sample intervals. Hence, the diameter of $K$ is large enough that it can contain two sample points but not three sample points. We express this relationship by requiring that

$$x \in K_y \Rightarrow K_x \cap K_y \cap S \neq \emptyset$$
$$\text{and } K \cap S = \{0\}.$$

The first condition implies that the dilation of sample points fills the whole space; that is, $S \oplus K = E^N$ when $K$ is not empty. If the points in the sampling set $S$ are spaced no further than $d$ apart, then the corresponding reconstructing kernel $K$ could be the topologically open ball of radius $d$ where the norm used to define distance is the $L_\infty$ norm. In this case, $x \in K_y \Rightarrow K_x \cap K_y \cap S \neq \emptyset$. Notice that two points which are $d$ apart can lie on the diameter of $K$. But since the ball is topologically open, the diameter cannot contain 3 points spaced $d$ apart. Hence, the radius of $K$ is just smaller than the sampling inteval. Also notice that if a sample point falls in the center of $K$, $K$ will not contain another sample point.

We are now ready to state some propositions which lead to the binary morphological sampling theorem. In what follows, the set $F \subseteq E^N$, the reconstruction structuring element will be denoted by $K \subseteq E^N$, and the sampling set will be denoted by $S \subseteq E^N$. Although not necessary for every relationship, we assume that $S$ and $K$ obey the following five conditions:

$$(1) S = S \oplus S,$$
$$(2) S = \check{S},$$
$$(3) K \cap S = \{0\},$$
$$(4) K = \check{K},$$
$$(5) a \in K_b \Rightarrow K_a \cap K_b \cap S \neq \emptyset.$$

*Figure 2 illustrates the $S$ associated with a 3 to 1* downsampling. Figure 3 illustrates a structuring element $K$ satisfying (3), (4) and (5). Since the dilation operation is commutative and associative, conditions (1) through (3) imply that the sampling set $S$ with the dilation operation comprises an abelian group with the origin being its unit element. Thus, if $x \in S$, then $S_x = S$, and also since $K \cap S = \{0\}$, $x \in S$ implies $K_x \cap S = \{x\}$. Both these facts are utilized in a number of the proofs to follow.

### 5.1 The Set Bounding Relationships

It is obvious that since $0 \in K$, the reconstruction of a sampled set $F \cap S$ by dilation with $K$ produces a superset of the sampled set $F \cap S$. That is, $F \cap S \subseteq (F \cap S) \oplus K$. The reconstruction by dilation is open so that $[(F \cap S) \oplus K] \circ K = (F \cap S) \oplus K$. Moreover, the erosion and dilation of the original image $F$ by $K$ bound the reconstructed sampled image in the sense of $F \circ K \subseteq (F \cap S) \oplus K \subseteq F \oplus K$.

This first bounding relationship indicates that the reconstruction by dilation cannot be too far away from $F$ since the reconstruction is constrained to lie between $F$ eroded by $K$ and $F$ dilated by $K$. Our next relationship strengthens the closeness between $F$ and the dilation reconstruction $(F \cap S) \oplus K$. Sampling $F$ and sampling the dilation reconstruction of $F$ produce identical results: $F \cap S = [(F \cap S) \oplus K] \cap S$.

Considering sampling followed by the dilation reconstruction as an operation we discover that it is an increasing operation, distributes over union but not over intersection. That is:

$(1) F_1 \subseteq F_2$ implies $(F_1 \cap S) \oplus K \subseteq (F_2 \cap S) \oplus K$

$(2)\big((F_1 \cup F_2) \cap S\big) \oplus K = \big[(F_1 \cap S) \oplus K\big] \cup \big[(F_2 \cap S) \oplus K\big]$

$(3)\big((F_1 \cap F_2) \cap S\big) \oplus K \subseteq \big[(F_1 \cap S) \oplus K\big] \cap \big[(F_2 \cap S) \oplus K\big]$

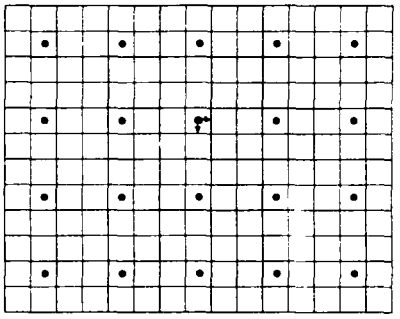


**Figure 2** *illustrates sampling every third pixel by row and by column. The sampling set $S$ is represented by all points which are shown as "•".*

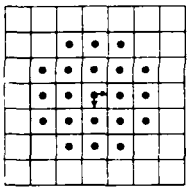


**Figure 3** *illustrates a symmetric structuring element $K$ which is a digital disc of radius $\sqrt{5}$. For the sampling set $S$ of Figure 2, $K \cap S = \{0\}$ and $x \in K_y$ implies $K_x \cap K_y \cap S \neq \emptyset$.*

Our next relationship states that the dilation reconstruction of a sampled $F$ is always a superset of F opened by the reconstruction structuring element $K$. Hence, if $F$ is open under $K$, then $F$ is contained in its dilation reconstruction: $F \circ K \subseteq (F \cap S) \oplus K$.

Thus the reconstruction of the opened sampled image $F$ is bounded by $F \circ K$ on the low side and $F \circ K$ dilated by $K$ on the high side.

$$F \circ K \subseteq \big[(F \circ K) \cap S\big] \oplus K \subseteq (F \circ K) \oplus K$$

If $F$ is morphologically simplified and filtered so that $F = F \circ K$, then the previous bounds reduce to

$$F \subseteq (F \cap S) \oplus K \subseteq F \oplus K$$

By reconsidering our example $F = (3.1,\ 7.4) \cup (11.5,\ 11.6) \cup (18.9,\ 19.8)$ which is not open under $K = (-1, 1)$, we can see that such an $F$ is not necessarily a lower bound for the reconstruction. In this case $F \cap S = \{4, 5, 6, 7, 19\}$ and the reconstruction $(F \cap S) \oplus K = (3, 8) \cup (18, 20)$, which does not contain $F$. This suggests that the condition that $F$ be open under $K$ is essential in order to have $F \subseteq (F \cap S) \oplus K$.

We now state one last relation between the reconstruction $(F \cap S) \oplus K$ and $F$. The reconstruction $(F \cap S) \oplus K$ is the largest open set which when sampled produces $F \cap S$.

Proposition

Let $A \subseteq E^N$ satisfy $A \cap S = F \cap S$ and $A = A \circ K$. Then $A \supseteq (F \cap S) \oplus K$ implies $A = (F \cap S) \oplus K$.

Proof

Suppose $A \supseteq (F \cap S) \oplus K$ and $A \cap S = F \cap S$ and $A = A \circ K$. Since $A \cap S = F \cap S$, $(A \cap S) \oplus K = (F \cap S) \oplus K$. But $A = A \circ K$ implies $A \subseteq (A \cap S) \oplus K = (F \cap S) \oplus K$. Now $A \subseteq (F \cap S) \oplus K$ together with the supposition $A \supseteq (F \cap S) \oplus K$ implies $A = (F \cap S) \oplus K$.

As the reconstruction $(F \cap S) \oplus K$ is maximal with respect to the two properties of being open and downsampling to $F \cap S$, we are naturally led to ask about a minimal reconstruction. Certainly we would expect a minimal reconstruction to be contained in the maximal reconstruction and contain the sampled image. Since closing is extensive, we immediately have $F \cap S \subseteq (F \cap S) \bullet K$. Since $0 \in K$, erosion is an anti-extensive operation. Hence, $(F \cap S) \bullet K = [(F \cap S) \oplus K] \ominus K \subseteq (F \cap S) \oplus K$. These relations suggest the possibility of a reconstruction by closing. Indeed a closing reconstruction has set bounds similar to the dilation reconstruction: $F \ominus K \subseteq (F \cap S) \bullet K \subseteq (F \cap S) \oplus K \subseteq F \oplus K$.

For true reconstruction, the sampled reconstruction should be identical to the sampled image. Indeed, this is the case $[(F \cap S) \bullet K] \cap S = F \cap S$.

Consider our example $F = (3.1,\ 7.4) \cup (11.5,\ 11.6) \cup (18.9,\ 19.8)$, which is closed under $K = (-1,\ 1)$. If the sampling set $S$ is the integers then $F \cap S = \{4, 5, 6, 7, 19\}$. Closing $F \cap S$ with $K$ can be visualized via the opening/closing duality $(F \cap S) \bullet K = ((F \cap S)^C \circ \check{K})^C$. Opening the set $(F \cap S)^C$ with $\check{K} = K$ produces $(F \cap S)^C \circ K = \{\ x \neq 19 \mid x < 4 \text{ or } > 7\}$. Hence $(F \cap S) \bullet K = ((F \cap S)^C \circ K)^C = \{x \mid x = 19 \text{ or } 4 \leq x \leq 7\}$, and sampling produces $[(F \cap S) \bullet K] \cap S = \{4, 5, 6, 7, 19\} = F \cap S$.

From the previous relationship, it rapidly follows that sampling followed by a reconstruction by closing is an idempotent operation. That is, $[(F \cap S) \bullet K] \cap S \bullet K = (F \cap S) \bullet K$. A reconstruction by closing is obviously closed under $K$. Moreover, it can be quickly determined that sampling followed by a closing reconstruction is increasing and does not necessarily distribute over union or intersection. That is,

$F_1 \subseteq F_2$ implies $(F_1 \cap S) \bullet K \subseteq (F_2 \cap S) \bullet K$

$[(F_1 \cup F_2) \cap S] \bullet K \supseteq [(F_1 \cap S) \bullet K] \cup [(F_2 \cap S) \bullet K]$

$[(F_1 \cap F_2) \cap S] \bullet K \subseteq [(F_1 \cap S) \bullet K] \cap [(F_2 \cap S) \bullet K]$

Furthermore, the closing reconstruction of a sampled $F$ is always a subset of $F$ closed by the reconstruction structuring element $K$. That is, $(F \cap S) \bullet K \subseteq F \bullet K$, so that $((F \bullet K) \cap S) \bullet K \subseteq F \bullet K$. Hence a closing reconstruction of an image which is closed before sampling will be a subset of the closed image.

By considering a simple example $F = \{0,1\}$, which is not closed under $K = (-1,1)$, we can see that $F$ is not necessarily an upper bound for the reconstruction. In this case, $F \cap S = \{0,1\} = F$ and the reconstruction $(F \cap S) \bullet K = F \bullet K = [0,1]$ which properly contains $F$. This suggests that the condition that $F$ be closed under $K$ is essential in order to have $(F \cap S) \bullet K \subseteq F$.

Finally, we state one last relation between the reconstruction $(F \cap S) \bullet K$ and $F$. The reconstruction $(F \cap S) \bullet K$ is the smallest closed set which when sampled produces $F \cap S$.

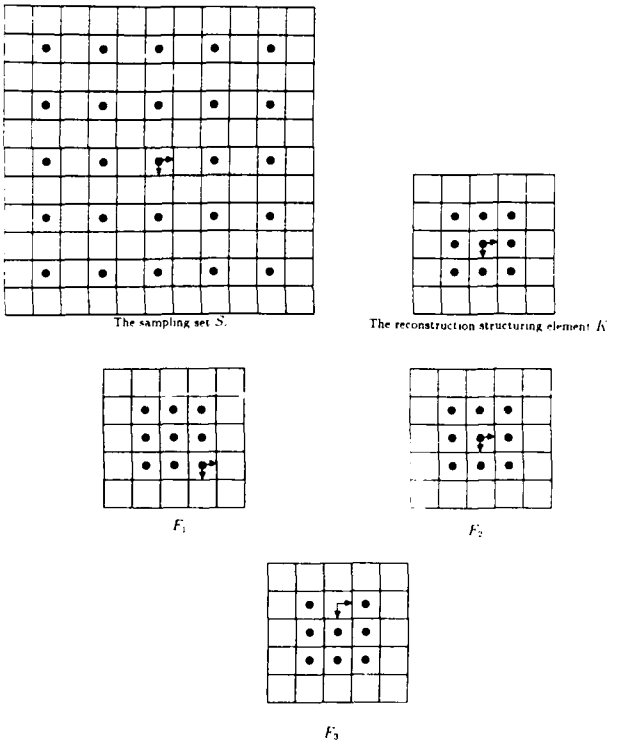



**Figure 4** *illustrates a sampling set $S$, a reconstruction structuring element $K$, and three sets, $F_1, F_2$, and $F_3$, each of which is open under $K$.*

### 5.2 Examples

To better illustrate the bounding relationships developed in the previous section between a set and its sample reconstructions, we show three simple examples. The domain of these examples is defined as $E \times E$ where $E$ is the set of integers. The sample set $S$ is chosen as the set of even numbers in both row and column directions. Thus,

$$S = \{(r,c) | r \in E \text{ and is even}; \quad c \in E \text{ and is even}\}.$$

$K$ is chosen as a box of size $3 \times 3$ whose center is defined as the origin. The sets $S, K$, and the three example sets $F_1, F_2$, and $F_3$ are shown in Figure 4. The sets $F_1, F_2$, and $F_3$ are $3 \times 3$ boxes having different origins and the condition $F = F \circ K$ holds for all these example sets.

The results of $F \ominus K, F \cap S, (F \cap S) \bullet K, (F \cap S) \oplus K$, and $F \oplus K$ for sets $F_1, F_2$ and $F_3$ are shown in Figures 5,6, and 7 respectively.

#### 5.2.1 Example 1

All the pixels contained in the vertical boundaries of $F_1$ have even column coordinates and those in the horizontal boundaries of $F_1$ have even row coordinates. Since the sample set $S$ consists of pairs of even numbers and $F_1$ is a $3 \times 3$ box, the set $F_1 \cap S$ consists of the four corner points of $F_1$ and is contained in the boundary set of $F_1$. Hence the closing reconstruction of $F_1 \cap S$ recovers $F_1$ and the dilation reconstruction of $F_1 \cap S$ is equivalent to $F \oplus K$. In fact, the following two equalities hold only when (1) the sampling is every other row and column, (2) a set's vertical boundaries have even column coordinates, and (3) its horizontal boundaries have even row coordinates

$$(F \cap S) \bullet K = F \text{ and}$$
$$(F \cap S) \oplus K = F \oplus K.$$

The bounding relationships for $F_1$, illustrated in Figure 5, are

$$F_1 \ominus K \subseteq (F_1 \cap S) \bullet K = F_1 \subseteq (F_1 \cap S) \oplus K = F_1 \oplus K.$$

#### 5.2.2 Example 2

Since all pixels contained in the vertical boundaries of $F_2$ have odd column coordinates and those in the horizontal boundaries of $F_2$ have odd row coordinates and $F_2$ is a small $3 \times 3$ box, the set $F_2 \cap S$ does not contain any part of the boundary of $F_2$. Thus the closing reconstruction of $F_2 \cap S$ equals $F_2 \ominus K$ and the dilation reconstruction of $F_2 \cap S$ is equivalent to $F_2$. Similar to the example 1, the following equalities hold only when the sampling is every other row and column and has its odd column coordinates in its vertical boundaries and its odd row coordinates in its horizontal boundaries.

$$F \ominus K = (F \cap S) \bullet K \text{ and}$$
$$F = (F \cap S) \oplus K.$$

The bounding relationships for $F_2$, illustrated in Figure 6, are

$$F_2 \ominus K = (F_2 \cap S) \bullet K \subseteq F_2 = (F_2 \cap S) \oplus K \subseteq F_2 \oplus K.$$

### 5.2.3 Example 3

The pixels contained in the vertical boundaries of $F_3$ have odd column coordinates and the pixels in the horizontal boundaries of $F_3$ have even row coordinates. Hence, no equalities should exist in the bounding relationship. This is illustrated in Figure 7. The bounding relationships for $F_3$ are

$$F_3 \ominus K \subseteq (F_3 \cap S) \bullet K \subseteq F_3 \subseteq (F_3 \cap S) \oplus K \subseteq F_3 \oplus K.$$

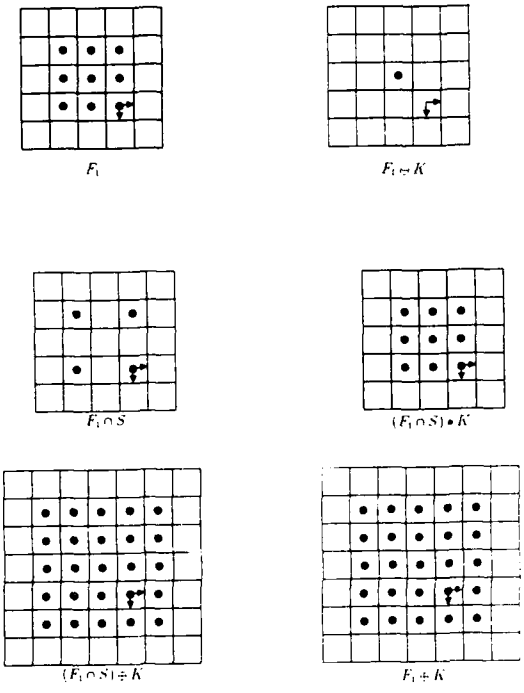



**Figure 5** *shows how the erosion and dilation of $F_1$ bound the minimal reconstruction $(F_1 \cap S) \bullet K$ and the maximal reconstruction $(F_1 \cap S) \oplus K$, respectively, which in turn bound $F_1$. because $F_1$ is both open and closed under $K$.*

To show why the opening condition $F = F \circ K$ is needed for the bounding relationships involving $F$, we show an example set $F_4$ which deviates from the set $F_3$ by adding six extra points to it (see Figure 8). The sample and reconstruction results of $F_4$, $F_4 \cap S$, $(F_4 \cap S) \bullet K$, and $(F_4 \cap S) \oplus K$ are exactly the same as the results for $F_3$. However, no bounding relationships between $F_4$ and its sample reconstructions are applicable. If we open $F_4$ by $K$, the bounding relationships exist because $F_4 \circ K = F_3$.

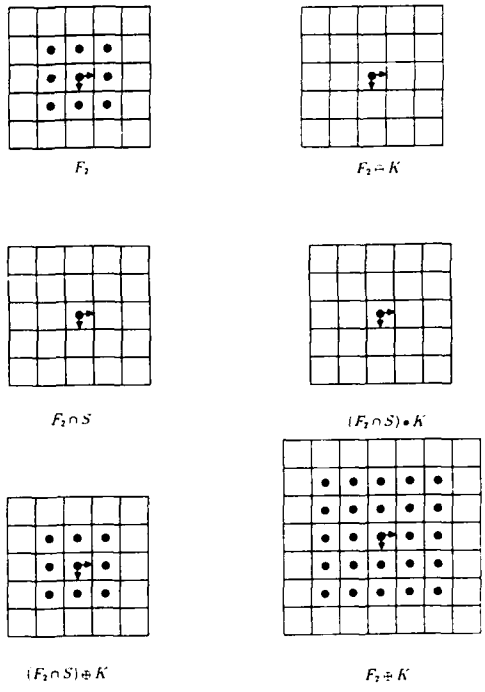



**Figure 6** *shows a second example of how the erosion and dilation of $F_2$ bound the minimal reconstruction $(F_2 \cap S) \bullet K$ and the maximal reconstruction $(F_2 \cap S) \oplus K$, respectively, which in turn bound $F_2$.*

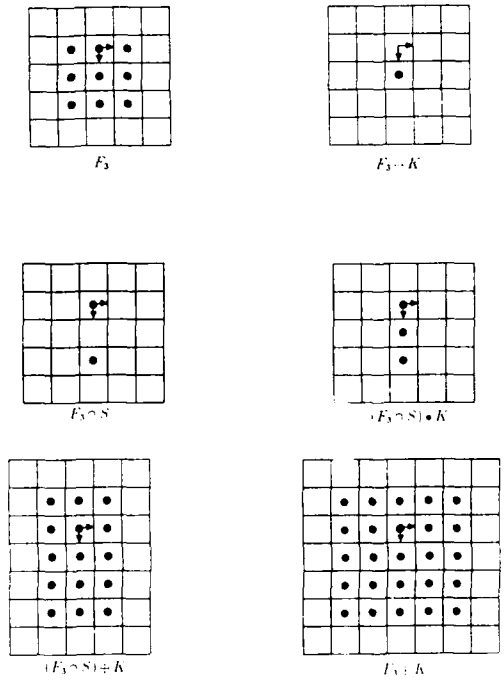
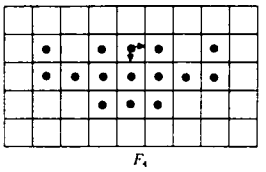
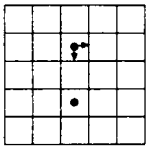



**Figure 7** *shows a third example of how erosion and dilation of $F_3$ bound (in this case properly) the minimal*

*reconstruction* $(F_3 \cap S) \bullet K$ *and the maximal reconstruction* $(F_3 \cap S) \oplus K$, *respectively, which in turn bound (in this case properly)* $F_3$.



$F_4$



$F_4 \cap S$

**Figure 8** *shows a set* $F_4$ *which is not open under* $K$. *Its sampling* $F_4 \cap S$ *is identical to the sampling of* $F_3$ *yet the maximal reconstruction* $(F_4 \cap S) \oplus K$ *does not constitute an upper bound for* $F_4$ *as in the previous examples.*

### 5.3 The Distance Relationships

Having established the maximality of the reconstruction $(F \cap S) \oplus K$ with respect to the property of being open and downsampling to $F \cap S$, and the minimality of the reconstruction $(F \cap S) \bullet K$ with respect to the property of being closed and downsampling to $F \cap S$, we now give a more precise characterization of how far $F \ominus K$ is from $F \bullet K$, how far $F \circ K$ is from $F \oplus K$, and how far $(F \cap S) \bullet K$ is from $(F \cap S) \oplus K$. This is important to know since $F \ominus K \subseteq (F \cap S) \bullet K \subseteq F$ when $F = F \bullet K$, and $F \subseteq (F \cap S) \oplus K \subseteq F \oplus K$ when $F = F \circ K$, and $(F \cap S) \bullet K \subseteq F \subseteq (F \cap S) \oplus K$ when $F = F \circ K = F \bullet K$. Notice that in all three cases the difference between the lower and the upper set bound is just a dilation by $K$. This motivates us to define a distance function to measure the distance between two sets and to work out the relation between the distance between a set and its dilation by $K$ with the size of the set $K$. In this section we show that with a suitable definition of distance, all these distances are less than the radius of $K$. Since $K$ is related to the sampling distance, all the above-mentioned distances are less than the sampling interval.

For the size of a set $B$, denoted by $r(B)$, we use the radius of its circumscribing disk. Thus, $r(B) = \min_{x \in B} \max_{y \in B} \|x - y\|$. The more mathematically correct forms of inf for min and sup for max may be substituted when the space $E$ is the real line. For a set $A$ which contains a set $B$, a natural pseudo-distance from $A$ to $B$ is defined by $\rho(A,B) = \max_{x \in A} \min_{y \in B} \|x - y\|$. This pseudo distance satisfies (1) $\rho(A,B) \geq 0$, (2) $\rho(A,B) = 0$ implies $A \subseteq B$, and (3)$\rho(A,C) \leq \rho(A,B) + \rho(B,C) + r(B)$. The asymmetric relation (2) is weaker than the corresponding metric requirement that $\rho(A,B) = 0$ if and only if $A = B$, and relation (3) is weaker than the metric triangle inequality.

The pseudo distance $\rho$ has a very direct interpretation. $\rho(A,B)$ is the radius of the smallest disk which when used as a structuring element to dilate $B$ produces a result which contains $A$.

Proposition

Let $disk\ (r) = \{x \mid \|x\| \leq r\}$ and $A, B \subseteq F^N$. Then $\max_{a \in A} \min_{b \in B} \|a - b\| = inf\{r | A \subseteq B \oplus\ disk(r)\}$.

Proof

Let $\rho_0 = \max_{a \in A} \min_{b \in B} \|a - b\|$ and $r_0 = inf\ \{r | A \subseteq B \oplus\ disk(r)\}$. Let $a \in A$ be given. Let $b_0 \in B$ satisfy $\|a - b_0\| = \min_{b \in B} \|a - b\|$. Now, $\rho_0 = \max_{x \in A} \min_{y \in B} \|x - y\| \geq \min_{b \in B} \|a - b\|$. Hence, $\rho_0 \geq \|a - b_0\|$ so that $a - b_0 \in\ disk(\rho_0)$. Now, $b_0 \in B$ and $a - b_0 \in\ disk(\rho_0)$ implies $a = b_0 + (a - b_0) \in B \oplus\ disk(\rho_0)$. Hence $A \subseteq B \oplus\ disk(\rho_0)$. Since $r_0 = inf\ \{r | A \subseteq B \oplus\ disk(r)\}$, $r_0 \leq \rho_0$. Suppose $A \subseteq B \oplus\ disk(r_0)$. Then $\max_{a \in A} \min_{b \in B \oplus\ disk(r_0)} \|a - b\| = 0$. Hence, $\max_{a \in A} \min_{b \in B} \min_{y \in\ disk(r_0)} \|a - b - y\| = 0$. But $\|(a - b) - y\| \geq \|a - b\| - \|y\|$. Therefore,

$$\begin{aligned} 0 &\geq \max_{a \in A} \min_{b \in B} \min_{y \in\ disk(r_0)} \|a - b\| - \|y\| \\ &\geq \max_{a \in A} \min_{b \in B} \|a - b\| + \min_{y \in\ disk(r_0)} -\|y\| \\ &\geq \max_{a \in A} \min_{b \in B} \|a - b\| - \max_{y \in\ disk(r_0)} \|y\| \end{aligned}$$

Now $\rho_0 = \max_{a \in A} \min_{b \in B} \|a - b\|$ and $r_0 = \max_{y \in\ disk(r_0)} \|y\|$ implies $0 \geq \rho_0 - r_0$ so that $r_0 \geq \rho_0$. Finally, $r_0 \leq \rho_0$ and $r_0 \geq \rho_0$ implies $r_0 = \rho_0$.

The pseudo distance $\rho$ can be used as the basis for a true set metric by making it symmetric. We define the set metric $\rho_M(A,B) = \max\{\rho(A,B), \rho(B,A)\}$, also called the Hausdorf metric. $\rho_M$ satisfies $\rho_M(A,B) = inf\ \{r | A \subseteq B \oplus\ disk(r)$ and $B \subseteq A \oplus\ disk(r)\}$. This happens since

$$\begin{aligned} \rho_m(A,B) &= \max\{\rho(A,B), \rho(B,A)\} \\ &= \max\Big\{inf\{r | A \subseteq B \oplus\ disk(r)\}, \\ &\qquad inf\ \{r | B \subseteq A \oplus\ disk(r)\}\Big\} \\ &= inf\{r | A \subseteq B \oplus\ disk(r) \\ &\qquad \text{and } B \subseteq A \oplus\ disk(r)\} \end{aligned}$$

A strong relationship between the set distance and the dilation of sets must be developed to translate set bounding relationships to distance bounding relationships. We show that $\rho(A \oplus B, C \oplus D) \leq \rho(A,C) + \rho(B,D)$ and then quickly extend the result to $\rho_M(A \oplus B, C \oplus D) \leq \rho_M(A,C) + \rho_M(B,D)$.

Proposition

(1) $\rho(A \oplus B, C \oplus D) \leq \rho(A,C) + \rho(B,D)$

(2) $\rho_M(A \oplus B, C \oplus D) \leq \rho_M(A,C) + \rho_M(B,D)$

Proof

(1)

$$\begin{aligned}
\rho(A \oplus B, C \oplus D) &= \max_{x \in A \oplus B} \min_{y \in C \oplus D} \|x - y\| \\
&= \max_{a \in A} \max_{b \in B} \min_{c \in C} \min_{d \in D} \|a + b - c - d\| \\
&\leq \max_{a \in A} \max_{b \in B} \min_{d \in D} \min_{c \in C} [\|a - c\| + \|b - d\|] \\
&\leq \max_{a \in A} \max_{b \in B} \min_{d \in D} [(\min_{c \in C} \|a - c\|) + \|b - d\|] \\
&\leq \max_{a \in A} \min_{c \in C} \|a - c\| + \max_{b \in B} \min_{d \in D} \|b - d\| \\
&\leq \rho(A,C) + \rho(B,D)
\end{aligned}$$

(2)

$$\begin{aligned}
\rho_M(A \oplus B, C \oplus D) &= \max\{\rho(A \oplus B, C \oplus D), \\
&\qquad \rho(C \oplus D, A \oplus B)\} \\
&\leq \max\{\rho(A,C) + \rho(B,D), \rho(C,A) + \rho(D,B)\} \\
&\leq \max\{\rho(A,C), \rho(C,A)\} + \max\{\rho(B,D), \rho(D,B)\} \\
&\leq \rho_M(A,C) + \rho_M(B,D)
\end{aligned}$$

From this last result, it is apparent that dilating two sets with the same structuring element cannot increase the distance between the sets. Dilation tends to suppress differences between sets, making them more similar. More precisely, if $B = D = K$, then $\rho_M(A \oplus K, C \oplus K) \leq \rho_M(A,C)$. It is also apparent that $\rho_M(A, A \oplus K) = \rho_M(A \oplus \{0\}, A \oplus K) \leq \rho_M(A,A) + \rho_M(\{0\},K) = \rho_M(\{0\},K) \leq \max_{k \in K} \|k\|$. Indeed, since the reconstruction structuring element $K = \check{K}$ and $\emptyset \in K$, the radius of the circumscribing disk is precisely $\max_{k \in K} \|k\|$. Hence, the distance between $A$ and $A \oplus K$ is no more than the radius of the circumscribing disc of $K$.

Since $\rho_M(A, A \oplus K) \leq \max_{k \in K} \|k\|$ and $\max_{k \in K} \|k\| = r(K)$, we have $\rho_M(A, A \oplus K) \leq r(K)$. Also, since $A \bullet K \supseteq A$, $\rho_M(A \bullet K, A) = \rho(A \bullet K, A)$. Since $0 \in K$, $A \bullet K \subseteq A \oplus K$. Hence, $\rho_M(A \bullet K, A) = \rho(A \bullet K, A) \leq \rho\big((A \bullet K) \oplus K, A\big) = \rho(A \oplus K, A) \leq r(K)$.

From this, it immediately follows that the distance between the minimal and maximal reconstructions, which differ only by a dilation by $K$, is no greater than the size of the reconstruction structuring element; that is, $\rho_M((F \cap S) \bullet K, (F \cap S) \oplus K) \leq r(K)$. When $F = F \circ K = F \bullet K$, then $(F \cap S) \bullet K \subseteq F \subseteq (F \cap S) \oplus K$. Since the distance between the minimal and maximal reconstruction is no greater than $r(K)$ it is unsurprising that the distance between $F$ and either of the reconstructions is no greater than $r(K)$.

When the image $F$ is open under $K$, the distance between $F$ and its sampling $F \cap S$ can be no greater than $r(K)$. Why? It is certainly the case that $F \cap S \subseteq F \subseteq (F \cap S) \oplus K$. Hence $\rho_M(F, F \cap S) \leq \rho_M(F \cap S, (F \cap S) \oplus K) \leq r(K)$.

If two sets are both open under the reconstruction structuring element $K$, then the distance between the sets must be no greater than the distance between their samplings plus the size of $K$.

From this last result it is easy to see that if $F$ is closed under $K$, then the distance between $F$ and its minimal reconstruction $(F \cap S) \bullet K$ is no greater than $r(K)$. Consider,

$$\begin{aligned}
\rho_M\big(F, (F \cap S) \bullet K\big) &\leq \rho_M\big(F \cap S, \big((F \cap S) \bullet K\big) \cap S\big) \\
&\quad + r(K) \\
&= \rho_M(F \cap S, F \cap S) + r(K) = r(K)
\end{aligned}$$

These distance relationships mean that just as the standard sampling theorem cannot produce a reconstruction with frequencies higher than the Nyquist frequency, the morphological sampling theorem cannot produce a reconstruction whose positional accuracy is better than the radius of the circumscribing disk of the reconstruction structuring element $K$. Since the diameter of this disk is just short of being large enough to contain two sample intervals, the morphological sampling theorem cannot produce a reconstruction whose positional accuracy is better than the sampling interval.

### 5.4 Examples

We use the example sets $F_1$, $F_2$, $F_3$, and $F_4$ in computing the distance between the original images and the sample reconstruction images. The values $\max_{y \in K} \|x - y\|$ for each $x \in K$ are shown in Figure 9. The minimum value among them, $\sqrt{2}$, is the radius $r(K)$ since $r(K) = \min_{x \in K} \max_{y \in K} \|x - y\|$.

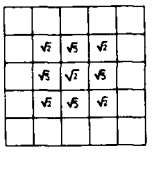



**Figure 9.** *The* $\max_{y \in K} \|x - y\|$ *values for all* $x \in K$*, where* $K$ *is the digital disc having radius* $\sqrt{2}$.

We now measure the distance between two sample reconstructions for all the example sets. To compute $\rho_M((F_1 \cap S) \bullet K, (F_1 \cap S) \oplus K)$ we first compute $\rho((F_1 \cap S) \oplus K, (F_1 \cap S) \bullet K)$ and $\rho((F_1 \cap S) \bullet K, (F_1 \cap S) \oplus K)$. The values $\min_{y \in (F_1 \cap S) \bullet K} \|x - y\|$ for all $x \in (F_1 \cap S) \oplus K$ are shown in Figure 10. The maximum value among them, $\sqrt{2}$, is the distance $\rho((F_1 \cap S) \oplus K, (F_1 \cap S) \bullet K)$. Similarly, we can compute $\rho((F_1 \cap S) \bullet K, (F_1 \cap S) \oplus K)$ which equals 0. Thus, $\rho_M((F_1 \cap S) \bullet K, (F_1 \cap S) \oplus K)$ equals $\sqrt{2}$ which is exactly the radius $r(K)$. Similarly, the distance between two reconstructions for sets $F_2$, $F_3$, and $F_4$ can be measured and they are all equal to $r(K)$.

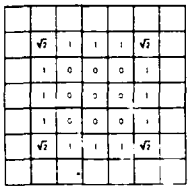


**Figure 10.** *The* $\min_{y \in (F_1 \cap S) \bullet K} \|x - y\|$ *for all* $x \in (F_1 \cap S) \oplus K$.

What is the distance $\rho_M(F, (F \cap S) \oplus K)$ for the example sets? Since $F_1 = (F_1 \cap S) \bullet K$, $\rho_M(F_1, (F_1 \cap S) \oplus K) = \rho_M((F_1 \cap S) \bullet K, (F_1 \cap S) \oplus K) = r(K)$. It is easy to see $\rho_M((F_2, (F_2 \cap S) \oplus K) = 0$ because $F_2 = (F_2 \cap S) \oplus K$. Figure 12 shows the values $\min_{y \in F_3} \|x - y\|$ for all $x \in (F_3 \cap S) \oplus K$, their maximum value being $\rho((F_3 \cap S) \oplus K, F_3) = 1$ Since $F_3 \subseteq (F_3 \cap S) \oplus K, \rho(F_3, (F_3 \cap S) \oplus K)$ equals 0. Hence, $\rho_M((F_3 \cap S) \oplus K, F_3) = 1 < r(K)$.

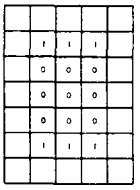


**Figure 11.** *shows* $\min_{y \in F_3} \|x - y\|$ *for each* $x \in (F_3 \cap S) \oplus K$.

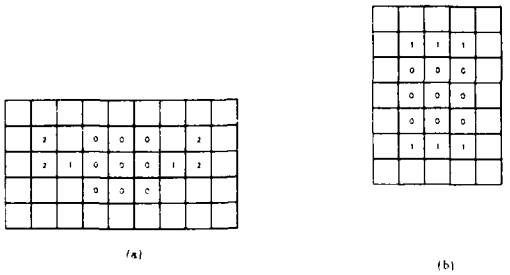


**Figure 12(a)** *shows values for* $\min_{y \in (F_4 \cap S) \oplus K} \|x - y\|$ *for all* $x \in F_4$. **Figure 12(b)** *shows values for* $\min_{y \in F_4} \|x - y\|$ *for all* $x \in (F_4 \cap S) \oplus K$. *The maximum among all these values is 2. Hence* $\rho_M(F_4 \cap S) \oplus k) = 2 > r(K)$.

The distance $\rho(F_4, (F_4 \cap S) \oplus K)$ is interesting since $F_4 \neq F_4 \circ K$. The $\min_{y \in (F_4 \cap S) \oplus K} \|x - y\|$ values for all $x \in F_4$ are shown in Figure 12a, their maximum value being $\rho(F_4, (F_4 \cap S) \oplus K) = 2$. The $\min_{y \in F_4} \|x - y\|$ values for all $x \in (F_4 \cap S) \oplus K$ are shown in Figure 12(b), the maximum value is $\rho((F_4 \cap S) \oplus K, F_4) = 1$ Thus, the distance $\rho_M(F_4, (F_4 \cap S) \oplus K)$ is equal to 2 which is greater than $r(K)$. This shows why the condition $F = F \circ K$ is required to bound the difference between $F$ and its maximum reconstruction $(F \cap S) \oplus K$. Similarly, we find

$$\rho_M((F_1 \cap S) \oplus K, F_1 \oplus K) = 0 < r(K)$$
$$\rho_M((F_2 \cap S) \oplus K, F_2 \oplus K) = \sqrt{2} = r(K)$$
$$\rho_M((F_3 \cap S) \oplus K, F_3 \oplus K) = 1 < r(K)$$
$$\rho_M((F_4 \cap S) \oplus K, F_4 \oplus K) = 2 > r(K)$$

Note that since $F_4 \neq F_4 \circ K$, $\rho_M((F_4 \cap S) \oplus K, F_4 \oplus K) \not\leq r(K)$. Using the minimum reconstruction, the positional accuracy for the example sets are

$$\rho_M(F_1, (F_1 \cap S) \bullet K) = 0 < r(K)$$
$$\rho_M(F_2, (F_2 \cap S) \bullet K) = \sqrt{2} = r(K)$$
$$\rho_M(F_3, (F_3 \cap S) \bullet K) = 1 < r(K)$$
$$\rho_M(F_4, (F_4 \cap S) \bullet K) = 3 > r(K)$$

Also, since $F_4 \neq F_4 \bullet K$, $\rho_M(F_4, (F_4 \cap S) \bullet K) \not\leq r(K)$.

### 5.5 Binary Morphological Sampling Theorem

This section summarizes the results developed in the previous sections. These results constitute the binary digital morphological sampling theorem.

Let $F, K, S \subseteq E^N$. Suppose $K$ and $S$ satisfy the sampling conditions

(1) $S \oplus S = S$
(2) $S = \check{S}$
(3) $K \cap S = \{0\}$
(4) $K = \check{K}$
(5) $x \in K_y$ implies $K_x \cap K_y \cap S \neq \emptyset$

Then
(1) $F \cap S = [(F \cap S) \bullet K] \cap S$
(2) $F \cap S = [(F \cap S) \oplus K] \cap S$
(3) $(F \cap S) \bullet K \subseteq F \bullet K$
(4) $F \circ K \subseteq (F \cap S) \oplus K$
(5) If $F = F \circ K = F \bullet K$, then $(F \cap S) \bullet K \subseteq F \subseteq (F \cap S) \oplus K$
(6) If $A = A \bullet K$ and $A \cap S = F \cap S$, then $A \subseteq (F \cap S) \bullet K$ implies $A = (F \cap S) \bullet K$
(7) If $A = A \circ K$ and $A \cap S = F \cap S$, then $A \supseteq (F \cap S) \oplus K$ implies $A = (F \cap S) \oplus K$
(8) If $F = F \bullet K$, then $\rho_M\big(F, (F \cap S) \bullet K\big) \leq r(K)$
(9) If $F = F \circ K$, then $\rho_M\big((F \cap S) \oplus K, F\big) \leq r(K)$

## 6. Operating In the Sampled Domain

Section 5 established the relationship between the information contained in the sampled set and the information contained in the unsampled set. It shows that a minimal and maximal reconstruction can be computed from the sampled

set. When the set is smooth enough with respect to the sampling $S$ (that is, when the set is both open and closed under the reconstruction structuring element), then the minimal and maximal reconstructions bound the unsampled set, differing from it by no more than the sampling interval length.

Not addressed in Section 5 is the relationship between the computationally more efficient procedure of morphologically operating in the sampled domain versus the less computationally efficient procedure of morphologically operating in the unsampled domain. In this section we quantify just exactly how close a morphological operation in the sampled domain can come to the corresponding morphological operation in the original domain. Thus we answer the question of how to compute the largest length of sampling interval which can produce an answer close enough to the desired answer when morphologically operating in the sampled domain.

The first proposition states that a sampled dilation contains the dilation of the sampled sets and a sampled erosion is contained in the erosion of the sampled sets.

Proposition

Let $B \subseteq E^N$ be the structuring element employed in the dilation or erosion. Then

(1) $(F \cap S) \oplus (B \cap S) \subseteq (F \oplus B) \cap S$

(2) $(F \cap S) \ominus (B \cap S) \supseteq (F \ominus B) \cap S$

Unfortunately, the containment relations cannot, in general, be strengthened to equalities. But we can determine the conditions under which the equality occurs and we can determine the distance between sets such as $(F \cap S) \oplus (B \cap S)$, which is the dilation of the sampled sets, and $(F \oplus B) \cap S$, which is the sampling of the dilation. In the sampled domain, we can compare the scheme of sampling and then performing the dilation in the sampled domain to dilating first and then sampling. We also inquire about how different things could be in the unsampled domain by comparing performing the dilation in the sampled space and then reconstructing versus performing the dilation in the unsampled domain. The next proposition states that this difference in the sampled domain cannot be more than $2r(K)$.

Proposition

If $F = F \circ K$ and $B = B \circ K$, then $\rho_M\Big((F \oplus B) \cap S, (F \cap S) \oplus (B \cap S)\Big) \leq 2r(K)$

Proof

First consider $\rho\Big((F \oplus B) \cap S, (F \cap S) \oplus (B \cap S)\Big) \leq \rho\Big(F \oplus B, (F \cap S) \oplus (B \cap S)\Big)$ Since $F = F \circ K$ and $B = B \circ K$, $F \subseteq (F \cap S) \oplus K$ and $B \subseteq (B \cap S) \oplus K$. Hence,

$$\begin{aligned}\rho\Big(F \oplus B, (F \cap S) \oplus (B \cap S)\Big) \leq \\ \rho\Big((F \cap S) \oplus K \oplus (B \cap S) \oplus K, (F \cap S) \oplus (B \cap S)\Big) \\ \leq \rho\Big([(F \cap S) \oplus (B \cap S)] \oplus K \oplus K, (F \cap S) \oplus (B \cap S)\Big) \\ \leq r(K \oplus K) \leq 2r(K)\end{aligned}$$

Next note that $\rho\Big((F \cap S) \oplus (B \cap S), (F \oplus B) \cap S\Big) = 0$. Since $(F \cap S) \oplus (B \cap S) \subseteq (F \oplus B) \cap S$. Now $\rho_M\Big((F \oplus B) \cap S, (F \cap S) \oplus (B \cap S)\Big) = \max\Big\{\rho\Big((F \oplus B) \cap S, (F \cap S) \oplus (B \cap S)\Big), \rho\Big((F \cap S) \oplus (B \cap S), (F \oplus B) \cap S\Big)\Big\} \leq \max\{2r(K), r0\} = 2r(K)$

Whereas dilation tends to suppress differences, erosion tends to accentuate differences. Consider the following example. Let $F$ be a disk of radius 12 and $B$ be a disk of radius 10. Then $F \ominus B$ is a disk of radius 2. Now define $F'$ to be a disk of radius 12 with its center point deleted. Notice that the pseudo set distance between $F$ and $F'$ is zero. But although $F'$ close to $F$, $F' \ominus B = \emptyset$. The difference of one point makes all the difference.

More formally, consider the difference between the erosion of $F$ and the erosion of $F \oplus K$.

$$\begin{aligned}\rho_M\Big((F \oplus K) \ominus B, F \ominus B\Big) &= \rho\Big((F \oplus K) \ominus B, F \ominus B\Big) \\ &\geq \rho\Big((F \ominus B) \oplus K, F \ominus B\Big)\end{aligned}$$

since $(F \oplus K) \ominus B \subseteq (F \ominus B) \oplus K$ where $\rho((F \ominus B) \oplus K, F \ominus B)$ *is no greater than and could be as close to* $r(K)$ as possible.

Thus we cannot expect that the difference between performing an erosion in the sampled domain versus performing a sampling of the erosion in the unsampled domain is no greater than the size of $K$. However, we do obtain set bounding relationships for dilation and erosion using the following relationships:

Dilating (eroding) a sampled set by a sampled structuring element is equivalent to sampling the dilation (erosion) of the unsampled set by the sampled structuring element.

(1) $(F \cap S) \oplus (B \cap S) = [F \oplus (B \cap S)] \cap S$

(2) $(F \cap S) \ominus (B \cap S) = [F \ominus (B \cap S)] \cap S$

Also, the dilation of the minimal reconstruction by a structuring element $B$ open under $K$ is contained in the dilation of the maximal reconstruction by the sampled structuring element $B \cap S$.

Lemma

Let $B = B \circ K$. Then $[(F \cap S) \bullet K] \oplus B \subseteq [(F \cap S) \oplus K] \oplus (B \cap S)$

Proof

Let $x \in [(F \cap S) \bullet K] \oplus B$. Then there exists an $f \in (F \cap S) \bullet K$ and $b \in B$ such that $x = f + b$. Since $B = B \circ K$, $b \in B$ implies there exists a $y$ such that $b \in K_y \subset B$. But because of the sampling constraint between $K$ and $S$, $b \in K_y$ implies $K_b \cap K_y \cap S \neq \emptyset$.

Therefore, there exists a $z \in K_b \cap K_y \cap S$. Now $z \in K_b$ implies that $z = k + b$ for some $k \in K$. Since it is also the case that $z \in K_y$, it must be that $z \in B$ because $K_y \subseteq B$. Recall that $x = f + b = f + z - k = (f - k) + z$. Since $f \in (F \cap S) \bullet K = [(F \cap S) \oplus K] \ominus K$ and since $-k \in \check{K} = K, f - k \in \Big([(F \cap S) \oplus K] \ominus K\Big) \oplus K = (F \cap S) \oplus K$. Since $z \in B$ and $z \in S, z \in B \cap S$. Finally, $f - k \in (F \cap S) \oplus K$ and $z \in B \cap S$ imply $x = (f - k) + z \in [(F \cap S) \oplus K] \oplus (B \cap S)$.

Now we see that dilation in the sampled domain and dilation in the unsampled domain are equivalent exactly when the structuring element $B$ of the dilation is open under $K$, and when the image $F$ is its minimal reconstruction.

Theorem

Let $B = B \circ K$. Then $(F \cap S) \oplus (B \cap S) = \{[(F \cap S) \bullet K] \oplus B\} \cap S$

Proof

$(F \cap S) \oplus (B \cap S) = \Big((F \cap S) \oplus B\Big) \cap S$ is always true. Since $F \cap S \subseteq (F \cap S) \bullet K$, $\Big((F \cap S) \oplus B\Big) \cap S \subseteq \{[(F \cap S) \bullet K] \oplus B\} \cap S$. But $[(F \cap S) \bullet K] \oplus B \subseteq [(F \cap S) \oplus K] \oplus (B \cap S)$ when $B = B \circ K$. Hence $(F \cap S) \oplus (B \cap S) \subseteq \{[(F \cap S) \bullet K] \oplus B\} \cap S \subseteq \{[(F \cap S) \oplus K] \oplus (B \cap S)\} \cap S$. Now $\{[(F \cap S) \oplus K] \oplus (B \cap S)\} \cap S = \{[(F \cap S) \oplus K] \cap S\} \oplus (B \cap S)$. Since $[(F \cap S) \oplus K] \cap S = F \cap S$ always holds under the sampling conditions, there results $(F \cap S) \oplus (B \cap S) \subseteq [(F \cap S) \bullet K] \oplus (B \cap S) \subseteq (F \cap S) \oplus (B \cap S)$ so that $(F \cap S) \oplus (B \cap S) = [(F \cap S) \bullet K] \oplus (B \cap S)$.

The equality relationship established in the theorem immediately leads to a set bounding relationship for dilation.

$$[(F \ominus K) \oplus B] \cap S \subseteq \{[(F \cap S) \bullet K] \oplus B\} \cap S = (F \cap S) \oplus (B \cap S) \subseteq (F \oplus B) \cap S$$

Also from the theorem, it becomes apparent that the difference between the maximally reconstructed dilation and the dilation of the minimal reconstruction can be no more than the size of $K$ when $B$ is open under $K$. This happens because

$$\begin{aligned} &\rho_M([(F \cap S) \oplus (B \cap S)] \oplus K, [(F \cap S) \bullet K] \oplus B) \\ &\leq \rho_M(\{[(F \cap S) \oplus (B \cap S)] \oplus K\} \cap S, \\ &\quad \{[(F \cap S) \bullet K] \oplus B\} \cap S) + r(K) \\ &\leq \rho_M((F \cap S) \oplus (B \cap S), (F \cap S) \oplus (B \cap S)) + r(K) = r(K) \end{aligned}$$

Similarly, eroding a sampled image by a sampled structuring element is equivalent to eroding the maximal reconstruction by the structuring element and then sampling when the structuring element is open under $K$.

Theorem

Let $B = B \circ K$. Then $(F \cap S) \ominus (B \cap S) = \{[(F \cap S) \oplus K] \ominus B\} \cap S$

Proof

The sampling conditions imply $[(F \cap S) \oplus K] \cap S = F \cap S$. Hence,

$$\begin{aligned} (F \cap S) \ominus (B \cap S) &= \{[(F \cap S) \oplus K] \cap S\} \ominus (B \cap S) \\ &= \{[(F \cap S) \oplus K] \ominus (B \cap S)\} \cap S \\ &\supseteq \{[(F \cap S) \oplus K] \ominus B\} \cap S \end{aligned}$$

Under the sampling conditions, $(F \cap S) \ominus (B \cap S) \subseteq S$. So to complete the equality, we need to show that $(F \cap S) \ominus (B \cap S) \subseteq [(F \cap S) \oplus K] \ominus B$. Let $x \in (F \cap S) \ominus (B \cap S)$. Then $(B \cap S)_x \subseteq F \cap S$. Since $B = B \circ K$, $B \subseteq (B \cap S) \oplus K$. Hence $B_x \subseteq (B \cap S)_x \oplus K$. But $(B \cap S)_x \subseteq F \cap S$ s. that $B_x \subseteq (F \cap S) \oplus K$. Now by definition of erosion, if $B_x \subseteq (F \cap S) \oplus K$, then $x \in [(F \cap S) \oplus K] \ominus B$

This immediately leads to some set bounding relationships for erosion

$$\begin{aligned} &(F \ominus B) \cap S \subseteq (F \cap S) \ominus (B \cap S) = \\ &\qquad \{[(F \cap S) \oplus K] \ominus B\} \cap S \subseteq [(F \oplus K) \ominus B] \cap S \end{aligned}$$

The previous theorem also makes it apparent that the difference between the maximally reconstructed erosion and the erosion of the maximal reconstruction can be no more than the size of $K$ when both $B$ and the erosion of the maximal reconstruction are open under $K$. This happens because

$$\begin{aligned} &\rho_M\Big([(F \cap S) \ominus (B \cap S)] \oplus K, [(F \cap S) \oplus K] \ominus B\Big) \\ &\leq \rho_M\Big(\{[(F \cap S) \ominus (B \cap S)] \oplus K\} \cap S, \\ &\quad \{[(F \cap S) \oplus K] \ominus B\} \cap S\Big) + r(K) \\ &\leq \rho_M\Big((F \cap S) \ominus (B \cap S), (F \cap S) \ominus (B \cap S)\Big) + r(K) = r(K) \end{aligned}$$

Just as it was the case that dilating (eroding) a sampled set by a sampled structuring element is equivalent to sampling the dilation (erosion) of the unsampled set by the sampled structuring element, so it is also the case that opening (closing) a sampled set by a sampled structuring element is equivalent to sampling the opening (closing) of the unsampled set by the sampled structuring element. These relationships are useful in establishing when the opening and closing operation are equivalent in the sampled and unsampled domain.

(1) $[F \circ (B \cap S)] \cap S = (F \cap S) \circ (B \cap S)$

(2) $[F \bullet (B \cap S)] \cap S = (F \cap S) \bullet (B \cap S)$

The bounding relationships between the sampled and unsampled domains for the opening and closing operation now follow immediately.

Theorem

Suppose $B = B \circ K$, then

(1) $\{F \circ [(B \cap S) \oplus K]\} \cap S \subseteq (F \cap S) \circ (B \cap S) \subseteq \{[(F \cap S) \oplus K] \circ B\} \cap S$

(2) $\{[(F\cap S)\bullet K]\bullet B\}\cap S \subseteq (F\cap S)\bullet(B\cap S) \subseteq \{F\bullet[(B\cap S)\oplus K]\}\cap S$

Proof

(1) Notice that $[(B\cap S)\oplus K]\circ(B\cap S) = (B\cap S)\oplus K$. Under this condition, $\{F\circ[(B\cap S)\oplus K]\}\cap S \subseteq [F\circ(B\cap S)]\cap S$. But by a previous proposition $[F\circ(B\cap S)]\cap S = (F\cap S)\circ(B\cap S)$. Now suppose $x \in (F\cap S)\circ(B\cap S)$. Then there exists a $y$ such that $x \in (B\cap S)_y \subseteq F\cap S$. But $(B\cap S)_y \subseteq (F\cap S)$ implies $(B\cap S)_y \oplus K \subseteq (F\cap S)\oplus K$ since dilation is an increasing operation. Hence, $[(B\cap S)\oplus K]_y \subseteq (F\cap S)\oplus K$. Since $B = B\circ K$, $B \subseteq (B\cap S)\oplus K$. Then, $B_y \subset (F\cap S)\oplus K$. Also, $x \in (B\cap S)_y$ implies $x \in B_y$. Finally, $x \in B_y \subseteq (F\cap S)\oplus K$ implies $x \in [(F\cap S)\oplus K]\circ B$.

(2) By a previous proposition $(F\cap S)\bullet(B\cap S) = [F\bullet(B\cap S)]\cap S$. Since $[(B\cap S)\oplus K]\circ(B\cap S) = (B\cap S)\oplus K$, $[F\bullet(B\cap S)]\cap S \subseteq \{F\bullet[(B\cap S)\oplus K]\}\cap S$. Let $R = (F\cap S)\bullet K$. Since $B = B\circ K$, $R\oplus B$ is open under $K$. Hence $R\oplus B \subseteq [(R\oplus B)\cap S]\oplus K$. Now

$$(R\bullet B)\cap S = [(R\oplus B)\ominus B]\cap S \subseteq \Big(\{[(R\oplus B)\cap S]\oplus K\}\ominus B\Big)\cap S$$

But the sampled erosion of a maximal reconstruction is the erosion of the sampled set by the sampled structuring element. Hence,

$$\Big(\{[(R\oplus B)\cap S]\oplus K\}\ominus B\Big)\cap S = \Big([(R\oplus B)\cap S]\ominus(B\cap S)$$

And the sampled dilation of a minimal reconstruction is the dilation of the sampled set by the sampled structuring element. Hence,

$$[(R\oplus B)\cap S]\ominus(B\cap S) = [(R\cap S)\oplus(B\cap S)]\ominus(B\cap S)$$

Finally, $R\cap S = [(F\cap S)\bullet K]\cap S = F\cap S$ so that $\Big([(F\cap S)\bullet K]\bullet B\Big)\cap S \subseteq (F\cap S)\bullet(B\cap S)$.

The bounding relationships immediately imply the following equivalence for the opening and closing operations between the sampled and unsampled domains.

Theorem

Suppose $B = B\circ K$.

(1) If $F = (F\cap S)\oplus K$ and $B = (B\cap S)\oplus K$, then $(F\cap S)\circ(B\cap S) = (F\circ B)\cap S$

(2) If $F = (F\cap S)\bullet K$ and $B = (B\cap S)\oplus K$, then $(F\cap S)\bullet(B\cap S) = (F\bullet B)\cap S$

Proof

(1) If $F = (F\cap S)\oplus K$ and $B = (B\cap S)\oplus K$, the bounding relationship for opening becomes

$$(F\circ B)\cap S \subseteq (F\cap S)\circ(B\cap S) \subseteq (F\circ B)\cap S$$

from which we immediately obtain $(F\circ B)\cap S = (F\cap S)\circ(B\cap S)$

(2) If $F = (F\cap S)\bullet K$ and $B = (B\cap S)\oplus K$, the bounding relationship for closing becomes

$$(F\bullet B)\cap S \subseteq (F\cap S)\bullet(B\cap S) \subseteq (F\bullet B)\cap S$$

from which we immediately obtain $(F\bullet B)\cap S = (F\cap S)\bullet(B\cap S)$.

### 6.1 Examples

A simple example illustrates the bounding relationships of morphological operations operating in the pre- and post-sampled domain. The sample set $S$ and the set $K$ we used are those defined in the previous examples (see Figure 5). The sets $F$,$B$, and $K$ are defined in Figure 13. It is clear that $B = B\circ K$. In Figure 14, we show the results of down-sampling every other row and every other column, $F\cap S$, $B\cap S$, and the sampled domain morphological operations, $(F\cap S)\oplus(B\cap S)$, $(F\cap S)\ominus(B\cap S)$. The results $[(F\cap S)\bullet K]\oplus B$, $[(F\cap S)\oplus K]\ominus B$, $\{[(F\cap S)\bullet K]\oplus B\}\cap S$,and $\{[(F\cap S)\oplus K]\ominus B\}\cap S$ are shown in Figure 15. Note that the following equalities hold:

$$(F\cap S)\oplus(B\cap S) = \{[(F\cap S)\bullet K]\oplus B\}\cap S$$

and

$$(F\cap S)\ominus(B\cap S) = \{[(F\cap S)\oplus K]\ominus B\}\cap S.$$

Figure 16 shows $(F\oplus B)\cap S$, $(F\ominus B)\cap S$, $\Big(F\oplus(B\cap S)\Big)$, and $\Big(F\ominus(B\cap S)\Big)$. Note that the following are true:

$$(F\cap S)\oplus(B\cap S) \subseteq (F\oplus B)\cap S$$

and

$$(F\cap S)\ominus(B\cap S) \supseteq (F\ominus B)\cap S.$$

It can be easily verified that

$$(F\cap S)\oplus(B\cap S) = [F\oplus(B\cap S)]\cap S$$

and

$$(F\cap S)\ominus(B\cap S) = [F\ominus(B\cap S)]\cap S.$$

In practical multiresolution image processing applications we would like to perform morphological operations in the sampled domain to reduce the computational expense. How well can a morphological operation be performed in the sampled domain rather than the original domain can be answered by the relationships and distances between $(F\cap S)\oplus(B\cap S)$ and $(F\oplus B)\cap S$ as well as $(F\cap S)\ominus(B\cap S)$ and $(F\ominus B)\cap S$. Unfortunately, the distance

$$\rho_M((F\cap S)\oplus(B\cap S),(F\oplus B)\cap S) < 2r(K)$$

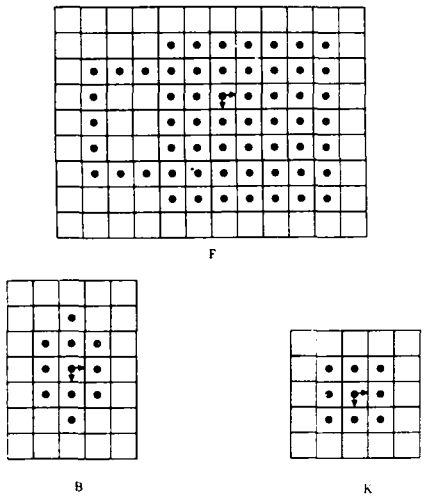



**Figure 13** *illustrates the sets $F$, $B$, and $K$.*

can be guaranteed only when $F = F \circ K$ and $B = B \circ K$. It can be very big when the set $F$ is not open. The set $F$ of Figure 13 is an example having a large difference between the pre- and post- sampled dilations because the conditions $F = F \circ K$ and $B = B \circ K$ are not satisfied.

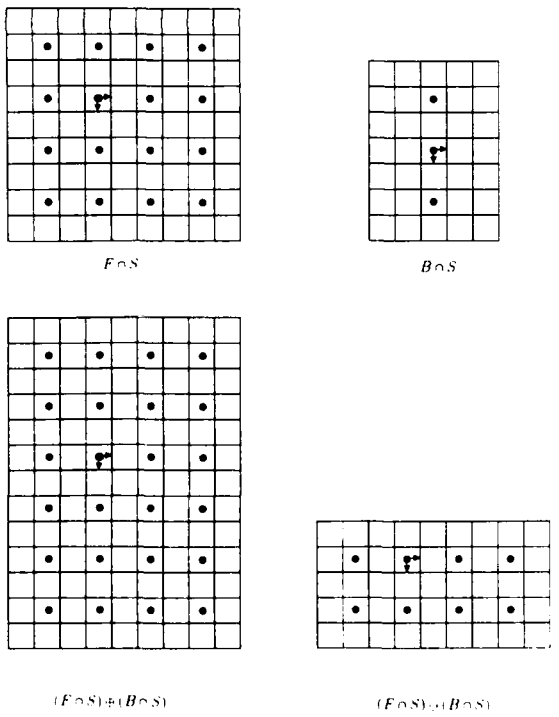



**Figure 14.** *shows the results of sampling the $F$ and $B$ of Figure 13 and performing the dilation and erosion of $F \cap S$ by $B \cap S$ in the sampled domain.*

We now show the distances between the pre- and post-sampled morphological operations. We first check the distance between $(F \cap S) \oplus (B \cap S)$ and $(F \oplus B) \cap S$. The $\min_{y \in (F \cap S) \oplus (B \cap S)} \|x - y\|$ values for all $x \in (F \oplus B) \cap S$ are shown in Figure 17; their maximum value is $\rho((F \oplus B) \cap S, (F \cap S) \oplus (B \cap S)) = 4$. Since $(F \cap S) \oplus (B \cap S) \subset (F \oplus B) \cap S$, the distance $\rho((F \cap S) \oplus (B \cap S), (F \oplus B) \cap S) = 0$. Thus, $\rho_M((F \cap S) \oplus (B \cap S), (F \oplus B) \cap S) = 4$. Note that

$$\rho_M((F \cap S) \oplus (B \cap S), (F \oplus B) \cap S) = 4 > 2r(K)$$

since $F \neq F \circ K$. Suppose $F' = F \circ K$ and $B' = B \circ K$. Figure 18 shows the results of $(F' \cap S) \oplus (B' \cap S)$ and $(F' \oplus B') \cap S$. Since $(F' \cap S) \oplus (B' \cap S) = (F' \oplus B') \cap S$ in this example, we find that

$$\rho_M((F' \cap S) \oplus (B' \cap S), (F' \oplus B') \cap S) = 0 < 2r(K).$$

Now we check the distances between the maximally reconstructed dilation (erosion) and the dilation (erosion) of the minimal (maximal) reconstruction, $\rho_M([(F \cap S) \oplus (B \cap S)] \oplus K, [(F \cap S) \bullet K] \oplus B)(\rho_M([(F \cap S) \ominus (B \cap S)] \oplus K, [(F \cap S) \oplus K] \ominus B))$. $[(F \cap S) \oplus (B \cap S)] \oplus K$ and $[(F \cap S) \bullet K] \oplus B$ are shown in Figure 19. The values of $\min_{y \in [(F \cap S) \bullet K] \oplus B} \|x - y\|$ for all $x \in [(F \cap S) \oplus (B \cap S)] \oplus K$ are shown in Figure 20; their maximum is $([(F \cap S) \oplus (B \cap S)] \oplus K, [(F \cap S) \bullet K] \oplus B) = 1$. Since $[(F \cap S) \bullet K] \oplus B \subseteq [(F \cap S) \oplus (B \cap S)] \oplus K$, this implies $\rho([(F \cap S) \bullet K] \oplus B, [(F \cap S) \oplus (B \cap S)] \oplus K) = 0$. Hence, $\rho_M([(F \cap S) \bullet K] \oplus B, [(F \cap S) \oplus (B \cap S)] \oplus K) =$

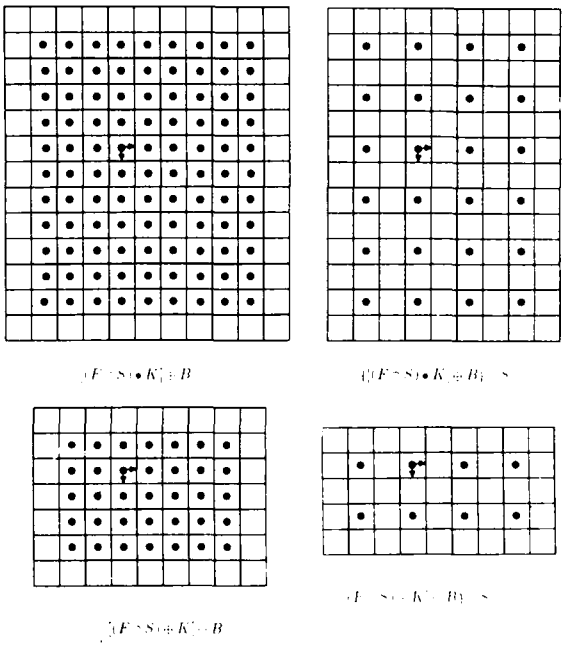


**Figure 15.** *shows the dilation and erosion of the minimal and maximal reconstruction of $F$ by the structuring element $B$ and also shows the sampling of this dilation and erosion.*

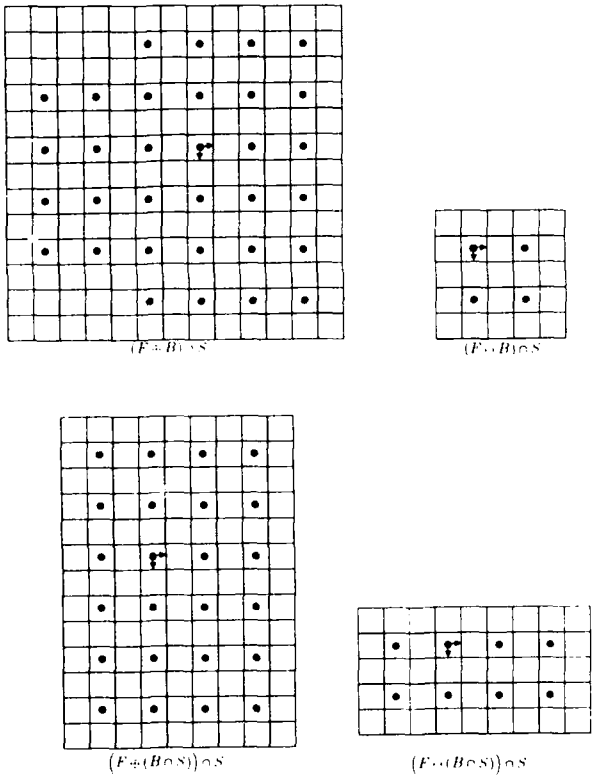


**Figure 16.** *shows some morphological operations in the original domain followed by sampling.*

$1 \le r(K)$. $[(F \cap S) \ominus (B \cap S)] \oplus K$ and $[(F \cap S) \oplus K] \ominus B$ are shown in Figure 20. Note that $[(F \cap S) \oplus K] \ominus B$ is open under $K$. The values of $\min_{y \in [(F \cap S) \oplus K] \ominus B} \|x - y\|$ for all $x \in [(F \cap S) \ominus (B \cap S)] \oplus K$ are shown in Figure 20; their maximum is $\rho([(F \cap S) \ominus (B \cap S)] \oplus K, [(F \cap S) \oplus K] \ominus B) = 1$. Since $[(F \cap S) \oplus K] \ominus B \subseteq [(F \cap S) \ominus (B \cap S)] \oplus K$, this implies $\rho([(F \cap S) \oplus K] \ominus B, [(F \cap S) \ominus (B \cap S)] \oplus K) = 0$. Hence, $\rho_M([(F \cap S) \oplus K] \ominus B, [(F \cap S) \ominus (B \cap S)] \oplus K) = 1 \le r(K)$.

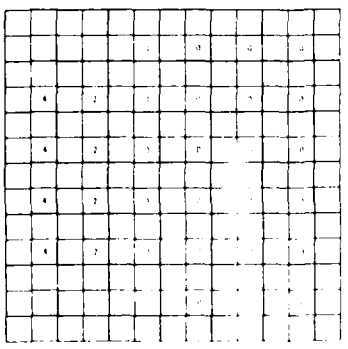


**Figure 17** *shows the values of* $\min_{y \in (F \cap S) \oplus (B \cap S)}$ *for all* $x \in (F \oplus B) \cap S$.

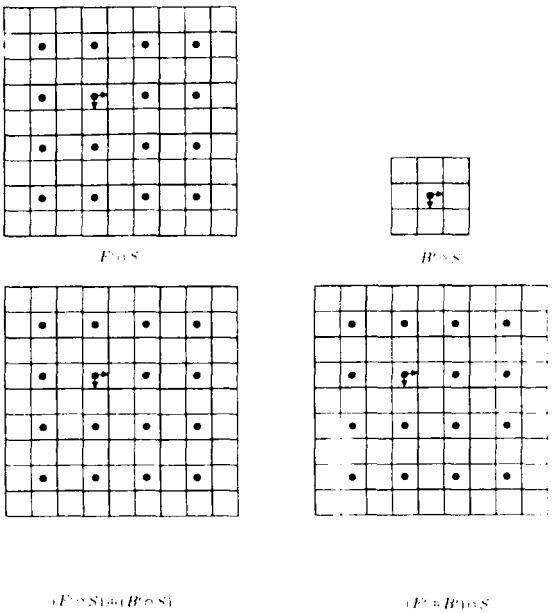


**Figure 18** *shows the results of* $(F' \cap S) \oplus (B' \cap S)$ *and where* $F' = F \circ K$ *and* $B' = B \circ K$. *(See Figure 14 for the definition of* $F, B$, *and* $K$.*)*

Next we show $f|_s \oplus k \le f \oplus k$. By the umbra homomorphism theorem, $U[f|_s \oplus k] = U[f|_s] \oplus U[k]$. But $U[f|_s] = U[f] \cap (S \times E)$. Hence, $U[f|_s \oplus k] = \{U[f] \cap (S \times E)\} \oplus U[k] \subseteq \big(U[f] \oplus U[k]\big) \cap \big((S \times E) \oplus U[k]\big)$. But $(S \times E) \oplus U[k] = E^N$ if and only if $S \oplus K = E^{N-1}$. So $U[f|_s \oplus k] \subseteq U[f] \oplus U[k] = U[f \oplus k]$ by the umbra homomorphism theorem. Hence, $T[U[f|_s \oplus k]] \le T[U[f \oplus k]]$ which by definition of gray scale dilation implies $f|_s \oplus k \le f \oplus k$. The analog of $F \cap S = [(F \cap S) \oplus K] \cap S$ is $f|_s = (f|_s \oplus k)|_s$. It holds under the condition that $k(0) = 0$.

In order to continue with the parallel development $f \circ k \le f|_s \oplus k$, we first need the stronger relation that for every $x \in F \circ K$, there exists a $s \in S \cap F$ such that $x \in K_s$ and $(f \circ k)(x) \le f(s) + k(x - s)$. This result follows from the sampling condition $u \in K_v$ implies $S \cap K_u \cap K_v \ne \emptyset$ and a constraint on the structuring element $k$ : $k(a) \le k(a - b) + k(b)$ for every $a, b$ satisfying $a \in K, b \in K$ and $a - b \in K$. This latter constraint is a new concept essential for the reconstruction structuring element in the gray scale morphology.

Before developing the proof for the inequality $(f \circ k)(x) \le f(s) + k(x - s)$ it will be useful to explore the meaning of the inequality $k(a) \le k(a - b) + k(b)$ since this is a constraint we have not had to deal with until now. The inequality $k(a) \le k(a - b) + k(b)$ together with $k = \check{k}$ implies that $k(y) \ge 0$ for every $y \in K$. This can easily be seen by letting $a = x + y$ and $b = x$. This leads to $k(x + y) \le k(x) + k(y)$. Then let $a = x$ and $b = x + y$. This leads to $k(x) \le k(y) + k(x + y)$. The two inequalities imply $k(x) \le k(y) + k(x + y) \le k(x) + 2k(y)$ from which $k(y) \ge 0$ quickly follows.

The inequality $k(a) \le k(a-b)+k(b)$ also implies that for any integer $n \ge 2, k(nx) \le nk(x)$ for every $x \in K$ satisfying $mx \in K$ for every $m,\ 2 \le m \le n$. The proof is by induction. Taking $n = 2, a = 2x$ and $b = x$ establishes the base case $k(2x) \le 2k(x)$. Suppose that for every $m, 2 \le m \le n, k(ma) \le mk(a)$ and $ma \in K$ and $a \in K$. Taking $a = (n+1)x$ and $b = x$ produces $k(n+1) \le k(nx)+k(x)$. But $k(nx) \le nk(x)$. Hence, $k\Big((n+1)x\Big) \le nk(x)+k(x) = (n+1)k(x)$. Now by induction $k(nx) \le nk(x)$ for every integer $n \ge 2$ satisfying $mx \in K$ for every $m, 2 \le m \le n$.

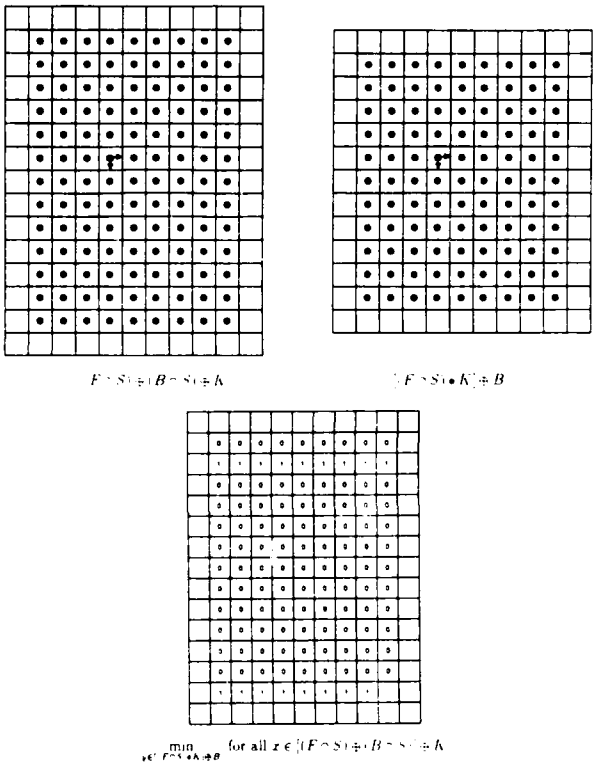


**Figure 19** *illustrates how the distance between the result produced by reconstructing the morphological dilation done in the sampled domain and the dilation of the minimal reconstruction done in the original domain must be less than* $r(K) = \sqrt{2}$.

## 7. The Grayscale Sampling Theorem

In this section we present the extension of the morphological sampling theorem from the binary case to the grayscale case.

### 7.1 The Grayscale Bounding Relationships

The set bounding relationships for the binary morphology have a direct correspondence to function bounding relationships in gray scale morphology. In this section we develop the bounding relationships without spending much time or discussion even though the extensions are somewhat more involved. The gray scale analog to the relationship $F \ominus K \subseteq (F \cap S) \oplus K \subseteq F \oplus K$ is $f \ominus k \le f|_s \oplus k \le f \oplus k$, where $f|_s : F \cap S \to E$ defined by $f|_s(x) = f(x)$ and it holds under very much the same conditions that the binary relationship holds. The only new requirement is for $k \ge 0$ which is stronger than the requirement that $0 \in U[k]$.

Finally, notice that $k(a) \le k(a-b)+k(b)$ for every $a, b \in K$ satisfying $a-b \in K$ and $K = \check{K}$ imply, as well, $k(b) \le k(b-a)+k(a)$ for every $a, b \in K$ satisfying $a-b \in K$. Since $k = \check{k}$, we obtain $k(a-b) \ge \max\{k(a)-k(b), k(b)-k(a)\}$.

Constructing functions which satisfy the inequality is easy with the following procedure. Define $k(0) = 0$ and $k(1)$ to be any positive number. Suppose $k(m)$, $m = 0, \ldots, n$ have been defined. Take $k(n+1)$ to be any number satisfying

$$\max_{1 \le v \le n} \{k(u)-k(n+1-u)\} \le k(n+1) \le \min_{1 \le v \le n} \{k(v)+k(n+1-v)\}$$

After $k$ is defined for all non-negative numbers in its domain define $k(-n) = k(n)$ for $n \ge 0$.

The generating procedure works because $k = \check{k}$ and the inequality implies $k(x)-k(y-x) \le k(y) \le k(x)+k(y-x)$. Hence, $\max_{1 \le v \le n} \{k(u)-k(y-u) \le k(y)\} \le \min_{1 \le v \le n} \{k(v)+k(y-v)\}$.

Proposition

Let $F, K, S \subseteq E^{N-1}, f : F \to E, k : K \to E$. Suppose $u \in K_v$ implies $S \cap K_u \cap K_v \ne \emptyset$ and $k(a) \le k(a-b)+k(b)$. Then for every $x \in F \circ K$, there exists an $s \in S \cap F$ such that $x \in K_s$ and $(f \circ k)(x) \le f(s)+k(x-s)$.

Proof

Let $x \in F \circ K$. Then $\Big(x, (f \circ k)(x)\Big) \in U[f \circ k]$ But $U[f \circ k] = U[f] \circ U[k]$. Hence, there exists $(u, v) \in E^{N-1} \times E$ such that $\Big(x, (f \circ k)(x)\Big) \in U[k]_{(u,v)} \subseteq U[f]$. Now $\Big(x, (f \circ k)(x)\Big) \in U[k]_{(u,v)}$ implies $\Big(x, (f \circ k)(x)\Big) - (u, v) \in U[k]$. So $\Big(x-u, (f \circ k)(x)-v\Big) \in U[k]$ which implies $(f \circ k)(x)-v \le k(x-u)$. Thus $v \ge (f \circ k)(x)-k(x-u)$. But $U[k]_{(u,v)} \subseteq U[f]$ implies for every $a \in K$ $\Big(a, k(a)\Big) + (u, v) \in U[f]$. Hence for every $a \in K, a+u \in F$ and $k(a)+v \le f(a+u)$. Now $x \in K_u$ implies there exists $s \in K_x \cap K_u \cap S$ so that $s-u \in K$. And since $s-u \in K, k(s-u)+v \le f(s-u+u) = f(s)$. Now $(f \circ k)(x)-k(x-u) \le v$ and $v \le f(s)-k(s-u)$ imply $(f \circ k)(x)-k(x-u) \le f(s)-k(s-u)$. But $k(a) \le k(a-b)+k(b)$ for every $a, b \in K$ satisfying $a-b \in K$. Letting $a = x-u$ and $b = s-u$, $a-b = x-s$, it is obvious that $x-u \in K$, $s-u \in K$, and $x-s \in K$. Hence, $k(x-u)-k(s-u) \le k\Big((x-u)-(s-u)\Big) = k(x-s)$. Therefore $(f \circ k)(x) \le f(s)+k(x-s)$.

Corollary:

Let $F, K, S \subseteq E^{N-1}$, $f : F \to E$, and $k : K \to E$. Suppose $u \in K_v$ implies $S \cap K_u \cap K_v \ne \emptyset$ and $k(a) \le k(a-b)+k(b)$. Then $(f \circ k)(x) \le (f|_s \oplus k)(x)$.

Having determined that $f \circ k \leq f|_s \oplus k$, the maximality of $f|_s \oplus k$ comes easily.

Proposition

Let $G, F, K, S \subseteq E^{N-1}$, $g : G \to E, k : K \to E, f : F \to E$. If $k(a) \leq k(a-b) + k(b)$ for every $a \in K$ and $b \in K$ satisfying $a - b \in K$, and $x \in K_y$ implies $K_x \cap K_y \cap S \neq \emptyset$, then $g = g \circ k, g|_s = f|_s$, and $g \geq f|_s \oplus k$ implies $g = f|_s \oplus k$.

We continue our development with the bounding relations for the minimal gray scale reconstruction.

Proposition

Let $F, K, S \subseteq E^{N-1}, f : F \to E$, and $k : K \to E$. If $k = \check{k}$, $k(a) \leq k(a-b) + k(b)$ for every $a, b \in K$ satisfying $a - b \in K$, and $x \in K_y$ implies $K_x \cap K_y \cap S \neq \emptyset$, then for every $u \in K$, $f(x+z) - k(z) \leq (f|_s \oplus k)(x+u) - k(u)$ for each $z \in K$.

Proof

Since $k(a) \leq k(a-b) + k(b)$, for every $a, b \in K$ satisfying $a - b \in K$, $-k(z) \leq -k(u) + k(u-z)$ for every $u, z \in K$ satisfying $u - z \in K$. Hence, $f(x+z) - k(z) \leq f(x+z) + k(u-z) - k(u)$ for every $u, z \in K$ satisfying $u - z \in K$. Making a change of variables $t = u - z$, there results

$$f(x+z) - k(z) \leq [f(x+u-t) + k(t)] - k(u), \text{ then}$$

$$f(x+z) - k(z) \leq [\max_{\substack{t \in K \\ x+u-t \in E \\ u-t \in K}} f|_s(x+u-t) + k(t)] - k(u)$$

$$\leq (f|_s \oplus k)(x+u) - k(u)$$

for every $u \in K$.

Proposition

Let $F, K, S \subseteq E^{N-1}$, $f : F \to E$, and $k : K \to E$. Suppose $u \in K_v$ implies $K_u \cap K_v \cap S \neq \emptyset$, $k = \check{k}$, and $k(a) \leq k(a-b) + k(b)$ for every $a, b \in K$ satisfying $a - b \in K$. Then for every $x \in F \ominus K$, $(f \ominus k)(x) \leq (f|_s \bullet k)(x)$.

Proof

Let $x \in F \ominus K$. Then $(f \ominus k)(x) = \min_{z \in K}\{f(x+z) - k(z)\} \leq (f|_s \oplus k)(x+u) - k(u)$ for every $u \in K$. There results, $(f \ominus k)(x) \leq \min_{u \in K}\{(f|_s \oplus k)(x+u) - k \bullet u)\} = (f|_s \bullet k)(x)$.

Proposition

Let $F, K, S \subseteq E^{N-1}$ and $f : F \to E$, $k : K \to E$. Then

(1) $f|_s = (f|_s \bullet k)|_s$.

(2) $f|_s \bullet k \leq f \bullet k$.

Proof

(1) Since closing is an increasing operation, $f|_s \leq f|_s \bullet k$. Since dilation is an increasing operation, $f|_s \bullet k \leq (f|_s \bullet k) \oplus k = f|_s \oplus k$. Hence, $f|_s \leq (f|_s \bullet k)|_s \leq (f|_s \oplus k)|_s$. But $(f|_s \oplus k)|_s = f|_s$ and this proves $f|_s = (f|_s \bullet k)|_s$.

(2) $U[f|_s \bullet k] = U[f|_s] \bullet U[k]$ by the umbra homomorphism theorem. Also, $U[f|_s] = U[f] \cap (S \times E)$ so that $U[f|_s] \subseteq U[f]$. Hence, $U[f|_s \bullet k] \subseteq U[f] \bullet U[k] = U[f \bullet k]$ by the umbra homomorphism theorem. This implies $f|_s \bullet k \leq f \bullet k$.

As before, the maximality comes easy.

Proposition

Let $G, F, K, S \subseteq E^{N-1}$, $g : G \to E$, $f : F \to E$, and $k : K \to E$. Suppose $g|_s = f|_s$ and $g = g \bullet k$. Then $g \leq f|_s \bullet k$ implies $g = f|_s \bullet k$.

### 7.2 The Grayscale Morphologic Sampling Theorem

This section summarizes the results developed in the previous sections. These results constitute the grayscale digital morphological sampling theorem.

Let $F, K, S \subseteq E^{N-1}$. Suppose $K$ and $S$ satisfy the sampling conditions

(1) $S \oplus S = S$

(2) $S = \check{S}$

(3) $K \cap S = \{0\}$

(4) $K = \check{K}$

(5) $x \in K_y$ implies $K_x \cap K_y \cap S \neq \emptyset$. Let $f : F \to E$ and $k : K \to E$. Suppose further that $k$ satisfies

(6) $k = \check{k}$

(7) $k(a) \leq k(a-b) + k(b)$ for every $a, b \in K$ satisfying $a - b \in K$

(8) $k(0) = 0$

Then

(1) $f|_s = (f|_s \bullet k)|_s$

(2) $f|_s = (f|_s \oplus k)|_s$

(3) $f|_s \bullet k \leq f \bullet k$

(4) $f|_s \circ k \leq f \circ k$

(5) If $f = f \circ k = f \bullet k$, then $f|_s \bullet k \leq f \leq f|_s \oplus k$

(6) If $g = g \bullet k$ and $g|_s = f|_s$, then $g \leq f|_s \bullet k$ implies $g = f|_s \bullet k$

(7) If $g = g \circ k$ and $g|_s = f|_s$, then $g \geq f|_s \oplus k$ implies $g = f|_s \oplus k$

### 7.3 Examples

The remaining examples illustrate how the morphological sampling theorem can lead to multiresolution processing techniques. The resolution hierarchy, called a pyramid, is produced typically by low pass filtering and then sampling to generate the next lower resolution level. Figure 21 shows a 5-level pyramid produced by pure sampling from a lasar radar range map. The highest resolution image size is 256 x 256. A 2-pix. wide line and a 4 x 4 box are placed intentionally at the upper right and upper left, respectively.

Figure 22 shows a 5-level morphological pyramid. At each level, the image is opened by a brick of size 3 x 3 and then sampled to generate the next lower resolution layer. Notice how the line in the upper right part of the image

has been eliminated. Figure 23 shows a similar 5-level morphological pyramid. In this pyramid the image at each level has been opened by a $3 \times 3$ brick, sampled, and then reconstructed, using the maximal reconstruction. The next lower resolution layer is generated by sampling as before.

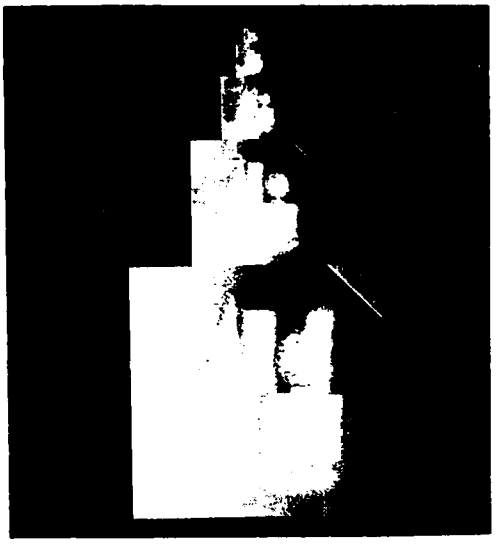


**Figure 21** *shows a 5-level pyramid of laser radar range image produced by pure sampling. The highest resolution image size is $256$ by 256. A 2 pixel wide line segment and a $4 \times 4$ box are intentionally placed at the upper right and upper left of the image, respectively.*

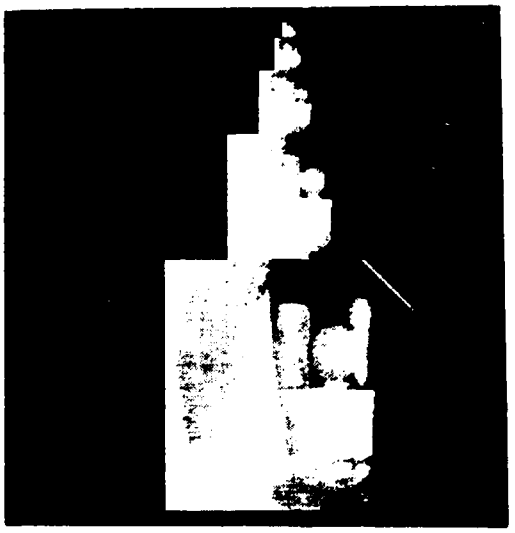


**Figure 22** *shows a 5-level pyramid of the same image as Figure 23, produced by, opened by a brick of $3 \times 3$ and then sampling to generate the next layer.*

**Figure 23** *shows a 5-level pyramid of the same image as Figure 23. In each level, the image is opened by a brick of $3 \times 3$ and then is sampled and reconstructed before it is down sampled to generate the next level.*

## 8. Research and Future Directions

The goal of our vision research group is to advance and develop the fundamental principles of computer vision by a systematic exploration of the required theoretical issues, the experimental issues, and the computer architecture issues. We do not approach theory for the sake of theory alone and we do not approach experiments for the sake of experiments and demonstrations alone. Rather, we employ statistically valid experiments using many images under a variety of conditions to explicitly or implicitly increase our understanding and insights about the constraints inherent in the reality with which computer vision deals. And we employ theory to build on the assumptions we currently hold about this validity. The technical material the morphological sampling theorem which we described in this paper establishes a sound basis for doing multi-resolution morphology. Sound basis means that we now understand what the coarsest resolution is for which a shape can be looked for and found. Once we find the shape at the coarsest resolution, we understand that the corresponding shape at the next highest resolution is guaranteed to differ from the extrapolated shape at the lower resolution level by no more than the sampling interval, anywhere. These guarantees play a crucial role in the high-level knowledge base of the vision system we are building.

Our current research in mathematical morphlogy deals with understanding how to do adaptive morphology and to automatically construct optimal morphological processing sequences on the basis of a training set of examples. We intend to first solve the processing sequence problem in the restricted domain where we only have to work with 2D shape. Then we will extend it to 2D perspective projections of 3D objects.

An active area of research is concerned with making a variety of vision algorithms robust in a quantifiable way and characterizing the performance of the robust algorithm. We believe that with a good enough characterization of the performance of each subalgorithm in a composite vision algorithm, we can develop the error propagation methodology to characterize the performance of the composite vision algorithm on the basis of the performance of its participating algorithms.

Robust here means making the algorithm perform almost the same whether or not there is some fraction of blunders or gross errors in the data. Not only do robust algorithms behave well, but they naturally indicate which of their inputs they estimate to be reliable and which inputs they estimate to be errorful.

The robust algorithms with which we have begun are line, curve and surface fitting, as well as the pose estimation problem, in which it is required to determine a sensor or object rotation and translation when corresponding data point sets are given. Four pose estimation problems arise from the situations in which the correspoding data point sets are 2D to 2D, 3D to 3D, and 2D perspective projection to 3D, 2D perspective projection to 2D perspective projection. The first case arises when flat manufactured objects are being viewed. The second case arises with data from range finder sensors. The third arises from 3D vision from monocular views, and the fourth arises in two view motion and stereo.

Once the robust algorithms for pose estimation are established and we have a performance characterization of how many pairs of corresponding points in a data set can be outright blunders with only a slight degradation in performance, there will be enough information to integrate the pose estimation directly into the matching that establishes the corresponding point data sets. In this manner we expect to be able to develop an overall theory and methodology by which integrated information transfer between algorithms at different levels of a vision paradigm can take place.

In the high level vision area we are continuing our earlier work on structrual matching via the consistent labeling framework as well as investigating inference algorithms which are appropriate for high level control and which are computationally efficient. The inference algorithms we are developing are extensions of the approximate linear time propagate and divide inference algorithms we developed for propositional logic. In the structural matching area, we are exploring how known structure on the constraints in a matching problem can be utilized to speed up the matching algorithm for either exact or inexact matching. In this area we have already demonstrated a polynomial time matcher suitable for industrial vision.

In the manufacturing area we are working on developing a methodology to automatically produce a vision algorithm given an augmented CAD data base of the object to be recognized or inspected. Our current efforts here have been to establish a representation for 3D objects suitable for vision algorithms and get this representation implemented in a CAD software system. We also have done some preliminary work to analytically predict what features we expect to find in an image of the object under different lighting sources and different viewing angles.

Our group is quite interested in parallel architectures for vision. We are exploring, designing, and building pyramid and reconfigurable pipeline architectures for iconic pixel pushing algorithms, the intermediate iconic/symbolic algorithms, as well as higher level symbolic algorithms. Some of the most challenging architectural issues arise at the intermediate level where we are focusing on several representative problems:

(1) determining where a given shaped object is;

(2) processing with chain codes or run length encoded data;

(3) Hough transforms with variable quantization and spatial clustering as a substitute for Hough transform;

(4) generating and computing using topological image structures such as the winged data structure for arc segments, and their relationships.

For reconfigurable pipeline architectures we have already developed a new language called INSIGHT which lets the programs specify the desired parallelism in a natural relational way. For the new architectural designs, we are working with other Electrical Engineering faculty whose main research areas are VLSI and parallel architectures.

## References


1. Narendra Ahuja and S. Swamy, "Multiprocessor pyramid architectures for bottom-up image analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.6, 1984, pp. 463-475.

2. Ken E. Batcher, "Design of a Massively Parallel Processor", *IEEE Transactions on Computers*, Vol. C-29, No. 9, September 1980, pp. 836-840.

3. P.J. Burt, "The pyramid as a structure for efficient computation," Ref. A. Rosenfeld, Ed., *Multiresolution Image Processing and Analysis*, Springer, Berlin, 1984, pp. 6-35.

4. E. North Coleman and Robert E. Sampson, "Acquisition of Randomly Oriented Workpieces Through Structure Mating", *Computer Vision Pattern Recognition Conference*, San Francisco, California, June 19-23, pp. 350-357.

5. T.R. Crimmins and W.M. Brown, "Image Algebra and Automation Shape Recognition," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-21, No.1, Jan, 1985, pp. 60-69.

6. J.L. Crowley, "A multiresolution representation for shape," Ref. A. Rosenfeld, Ed., *Multiresolution Image Processing and Analysis*, Springer, Berlin, 1984, pp. 169-189.

7. Edward Dougherty and Charles Giardina, *Image Processing-Continuous to Discrete, Volume I*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987.

8. Michael Duff, "Parallel Processors For Digital Image Processing", *Advances in Digital Image Processing*, (P. Stucki, Ed.), Plenum, New York, 1979, pp. 265-279.

9. Michael J.B. Duff, D.M. Watson, T.M. Fountain, and G.K. Shaw, "A Cellular Logic Array For Image Processing", *Pattern Recognition*, Vol. 5, 1973, pp. 299-247.

10. C.R. Dyer, "Pyramid algorithms and machines," Ref. K. Preston, Jr. and L. Uhr, Eds., *Multicomputers and Image Processing Algorithms and Programs*, Academic Press, New York, 1982, pp. 409-420.

11. Frans A. Gerritsen and L.G. Aardema, "Design and Use of DIP-1: A Fast Flexible and Dynamically Microprogrammable Image Processor", *Pattern Recognition*, Vol. 14, 1981, pp. 319-330.

12. Frans A. Gerritsen and Piet W. Verbeek, "Implementation of Cellular Logic Operators Using $3 \times 3$ convolution and Table Lookup Hardware", *Computer Vision, Graphics, and Image Processing*, Vol. 27, 1984, pp. 115-123.

13. M.J.E. Golay, "Hexagonal Parallel Pattern Transformations", *IEEE Transactions on Computers*, Vol. C-18, 1969, pp. 733-740.

14. D. Graham and P.E. Norgren, "The Diff3 Analyzer: A Parallel/Serial Golay Image Processor", *Real Time Medical Image Processing*, (Onoe, Preston, and Rosenfeld, Eds.), Plenum, London, 1980, pp. 163-182.

15. H. Hadw[illegible]esunger über Inhalt, Oberfläche und Isoperi[illegible] Springer, Berlin, 1957.

16. Robert M[illegible]aralick, "A Reconfigurable Systolic Network in Comput[illegible] Vision", *IEEE Computer Society Workshop on Computer Architecture For Pattern Analysis and Image Database Management*, Miami Beach, Florida, November 18-20, 1985, pp. 507-515.

17. Robert M. Haralick, Stanley R. Sternberg, and Xinhua Zhuang, "Image Analysis Using Mathematical Morphology," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No.4, July 1987, pp. 532-550.

18. M.J. Kimmel, R.S. Jaffe, J.R. Manderville, M.A. Lavin, "MITE: Morphic Image Transform Engine An Architecture For Reconfigurable Pipelines of Neighborhood Processors", *IEEE Computer Society Workshop on Computer Architecture For Pattern Analysis and Image Database Management*, Miami Beach, Florida, November 18-20, 1985,pp. 493-500.

19. J.C. Klein and J. Serra, "The Texture Analyzer", *Journal of Microscopy*, Vol. 95, 1977, pp. 349-356.

20. A. Klinger, "Multiresolution processing," Ref. A. Rosenfeld, Ed, *Multiresolution Image Processing and Analysis*, Springer, Berlin, 1984, pp.86- 100.

21. B. Kruse, "Design and Implementation of a Picture Processor", Science and Technology Dissertations, No. 13, University of Linkoeping, Linkoeping, Sweden, 1977.

22. James Lee, Robert M. Haralick, and Linda Shapiro, "Morphologic Edge Detection," *IEEE Journal of Robotics and Automation*, Vol. RA-3, No.1, April 1987, pp. 142-157.

23. Patrick F. Leonard, "Pipeline Architectures For Real-Time Machine Vision", *IEEE Computer Society Workshop on Computer Architecture For Pattern Analysis and Image Database Management*, Miami Beach, Florida, November 18-20, 1985, pp. 502-505.

24. R.M. Lougheed and D.L. McCubbery, "The Cytocomputer: A practical pipelined image processor," Proceedings of the 7th Annual International Symposium on Computer Architecture, 1980, pp. 1-7.

25. Petros Maragos and Ronald W. Schafer, "Morphological Filters—Part I: Their Set-Theoretic Analysis and Relations to Linear Shift-Invariant Filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP- 35, Aug, 1987, pp. 1153-1169.

26. Petros Maragos and Ronald W. Schafer, "Morphological Filters—Part II: Their Relations to Median, Order-Statistic, and Stack Filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-35, Aug, 1987, pp. 1170-1184.

27. G. Matheron, "Random Sets and Integral Geometry," Wiley, N.Y., 1975.

28. G. Matheron, *Elements Pour une Tiorre des Mulieux Poreux*, Masson, Paris, 1965.

29. David L. McCubbery and Robert M. Lougheed, "Morpholocial Image Analysis Using A Raster Pipeline Processor", *IEEE Computer Society Workshop on Computer Architecture For Pattern Analysis and Image Database Management*, Miami Beach, Florida, November 18-20, 1985, pp. 444-452.

30. R. Miller and Q.F. Stout, "Pyramid computer algorithms for determining geometric properties of images," Symposium on Computational Geometry, pp. 263-271.

31. H. Minkowski, "Volumen und Oberfläche", *Mathematical Annals*, Vol. 57, 1903, pp. 447-495.

32. A. Favre, R. Muggli, A. Stucki, and P. Bonderet, "Application of Morphologic Filters in the Assessment of Platelet Thrombus Deposition In Cross Sections of Blood Vessels", *4th Scandanavian Conference On Image Analysis*, Trondheim, Norway, June 17-20, 1985, pp. 629-640.

33. Shmuel Peleg, J. Naor, R. Hartley, and D. Avnir, "Multiple Resolution Texture Analysis and Classification", *IEEE Transaction Pattern Analysis and Machine Intelligence*, Vol. PAMI-60, 1984, pp. 518-523.

34. Shmuel Peleg and A. Rosenfeld, "A Min Max Medial Axis Transformation", *IEEE PAMI*, Vol. PAMI-3, No. 2, 1981, pp. 206-210.

35. J.L. Potter, "MPP Architecture and Programming", *Multicomputers and Image Processing*, (K. Preston and L. Uhr, Eds.), Academic Press, New York, 1982, pp. 275-290.

36. J.L. Potter, "Image Processing on the Massively Parallel Processor", *Computer*, Vol. 16, No. 1, January 1983, pp. 62-67.

37. William K. Pratt, "A Pipeline Architecture For Image Processing and Analysis". *IEEE Computer Society Workshop on Computer Architecture For Pattern Analysis and Image Database Management*, Miami Beach, Florida, November 18-20, 1985, pp. 516-520.

38. Ken Preston, Jr., "Machine Techniques for Automatic Identification of Binucleate Lymphocyte", *Proceedings Fourth International Conference on Medical Electronics*, Washington D.C., July 1961.

39. Ken Preston, Jr., "Application of Cellular Automata to Biomedical Image Processing", *Computer Techniques in Biomedicine and Medicine*, Auerbach Publishers, Philadelphia, 1973.

40. Azriel Rosenfeld, "Hierarchical representation: computer representations of digital images and objects," Ref. O.D. Faugeras, Ed. *Fundamentals in Computer Vision–An Advanced Course*, Cambridge University Press, Cambridge, UK, 1983, pp. 315-324.

41. J.Serra, "Stereology and structuring elements," *Journal of Microscopy*, , 1972, pp. 93-103.

42. Jean Serra, *Image Analysis and Mathematical Morphology*, Academic Press, Londion, 1982.

43. Stanley R. Sternberg, "Parallel Architectures For Image Processing". *Proceedings IEEE COMPSAC*, Chicago, 1979.

44. Stanley R. Sternberg, "Cellular Computers and Biomedical Image Processing", *Biomedical Images and Computers*, (J. Sklansky and J.C. Bisconte, Eds.), Springer Verlag, Berlin, 1982, pp. 294-319. (Presented at United States - France Seminar on Biomedical Image Processing, St. Pierre de Chartreuse, France, May 27-31, 1980.)

45. Stanley R. Sternberg, "Pipeline Architectures For Image Processing", *Multicomputers and Image Processing*, (K Preston and L. Uhr, Eds.), Academic Press, New York, 1982a, pp. 291-305.

46. Stanley R. Sternberg, "Esoteric Iterative Algorithms", *Second International Conference on Image Analysis and Processing*, Selva di Fasano, Brindisi, Italy, November 15-18, 1982b.

47. Stanley R. Sternberg, "Biomedical Image Processing", *Computer*, Vol. 16, No. 1, January 1983, pp. 22-34.

48. Steven L. Tanimoto, "Programming techniques for hierarchical parallel image processors," Ref. K. Preston, Jr. and L. Uhr, Eds., *Multicomputers and Image Processing Algorithms and Programs*, Academic Press, New York, 1982, pp. 431-429.

49. L. Uhr, "Pyramid multi-computer structures, and augmented pyramids, Ref. M.J.B. Duff, Ed., *Computing Structures for Image Processing*, Academic Press, London, 1983, pp. 95-112.

50. S.H. Unger, "A Computer Oriented to Spatial Problems", *Proceedings IRE*, Vol. 46, 1958, pp. 1744-1750.

51. Michael Werman and Shmuel Peleg, "Min-Max Operators In Texture Analysis", *IEEE PAMI*, Vol. PAMI-7, No. 6, Nov. 1985, pp. 730-733.

52. Stephen Wilson, "The Pixie-5000 - A Systolic Array Processor", *IEEE Computer Society Workshop on Computer Architecture For Pattern Analysis and Image Database Management*, Miami Beach, Florida, November 18-20, 1985, pp. 477-483.

53. A.P. Witkin, "Scale space filtering: a new approach to multi-scale description," Ref. Shimon Ullman and Whitman Richards, Eds., *Image Understanding*, Ablex Publishing Corp., Norwood, NJ, 1984, pp. 79-95.

54. Xinhua Zhuang and Robert M. Haralick, "Morphological Structuring Element Decomposition," *Computer Vision, Graphics, and Image Processing, 35*, (1986), pp. 370-382.

# Using Generic Geometric And Physical Models For Representing Solids*

Jean Ponce and Glenn Healey

Robotics Laboratory
Department of Computer Science
Stanford University
Stanford, California 94305

## Abstract

We present a solid modelling system based on powerful geometric and physical representations. The geometric representation combines constructive solid geometry (CSG) and boundary (Brep) representations. Solids are built as CSG trees by applying set operations to a very general set of primitives: generalized cylinders with either a straight or a curved axis. Using an efficient and robust boundary evaluator, we construct corresponding Breps in terms of trimmed surface patches and their boundaries. In addition to geometric models, our system describes the physical properties of material surfaces using very general physical models. Unlike many existing systems, we accurately model the optical properties of both homogeneous and inhomogeneous materials. Our model for the scattering of light from the body of inhomogeneous materials based on modified Kubelka-Munk theory is significantly more general than previous models used in computer graphics and vision for this process. This is a bit surprising, since this process is largely responsible for the appearance of the vast majority of the objects we see in everyday life. We unify our physical and geometric representations using a method to map material surface properties onto our geometric models. Our representations allow for computationally efficient and physically accurate display of complex solids. We present several new rendering algorithms exploiting these advantages, including fast ray tracing for line drawing and shaded display. The system described has been implemented. Several examples demonstrate the effectiveness of our geometric and physical models.

## Introduction

Being able to generate aesthethically pleasing images of a wide variety of objects requires expressive and robust geometric models and realistic yet possibly empirical models of the physical world. Conversely, being able to interpret complex images requires compact geometric models and a good understanding of the physics underlying the imaging process. In this work, we describe an implemented system which combines compact but expressive geometric models with accurate physical models of light sources and material surfaces.

The geometric modelling component of our system uses generalized cylinders as building blocks. We consider generalized cylinders with either a straight or a curved axis. They are well suited to visual tasks, as they combine both volume and surface information, and their shape is nicely "summarized" by their axis. These primitives are also very general, in particular, they include all the "natural" surfaces (cuboids, spheres, cylinders, cones, tori) used in most solid modelling systems. We have developed interactive tools for specifying the shape of generalized cylinders. Complex solids are built by combining these primitives through set operations. Our system allows this constructive solid geometry (CSG) representation to be built interactively, or from a simple modelling language. For many applications (e.g., prediction, display) however, it is necessary to have an explicit boundary representation (Brep) of solids. In the system presented here, it consists of a general description of objects' surfaces by graphs of trimmed surface patches separated by discontinuity curves. An efficient and robust boundary evaluator is used to transform the constructive solid geometry representation into the corresponding boundary representation. Powerful new rendering algorithms (including efficient ray tracing for line drawings and shaded display) have been developed which are capable of generating images of objects represented by our system. Figure 1 shows an example of such an object (a plastic valve).

We represent the physical properties of material surfaces using a generic physical modelling system. Our physical models are more general than any previously used models in several respects, yet these models are easy to use for image synthesis. Every parameter of our representation specifies an intrinsic physical property of a material surface. This is unlike many systems which model at the level of geometry dependent properties like reflectance. We represent a ma-

*Support for this work was provided in part by the Air Force Office of Scientific Research under contract F33615-85-C-5106 and by the Advanced Research Projects Agency of the Department of Defense under Knowledge Based Vision ADS subcontract S1093 S-1.

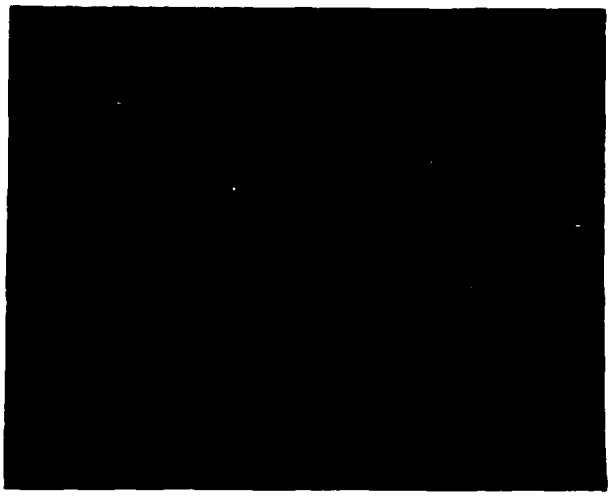


**Figure 1.** A plastic street elbow pipe modelled by our system.

**Figure 2.** The pipe of the previous Figure, cut to show its internal structure.

terial surface as the combination of a material description and a surface description. Physically, the material description specifies optical properties while the surface description specifies local geometric properties. Our models apply to both homogeneous and inhomogeneous materials. While most of the work on reflectance models in computer graphics has concentrated on specular reflection, we adopt here a new physical model for a process which is much more significant visually. This process is called colorant layer scattering and it is the dominant optical process for inhomogeneous materials (e.g. plastics, paints, textiles, paper, ceramics). The model we use has been verified experimentally by others and allows us to accurately predict variations in the color and intensity of the light scattered from the body of an inhomogeneous material as a function of geometry. Our system also includes general models for light sources. The physical properties (intensity and color) of the light reflected from surfaces are easy to compute and render using our physical models. We merge our geometric and physical representations for objects using functions which associate material and surface properties with regions on geometric models.

This paper concentrates on representational issues and the application of our models to graphics problems. Vision applications (prediction, segmentation, matching) are described elsewhere [Binford, Levitt, Mann, 1987], [Ponce, Chelberg, Mann, 1987], [Healey and Binford, 1987]. The paper is divided into three sections. Section 1 discusses our geometric models. Section 2 develops our physical models. Section 3 describes our rendering algorithms which use both our geometric and physical models. Examples are presented throughout to demonstrate the effectiveness of our geometric and physical models (Figures 1,2,4, 5,8).

## 1. Geometric Models

Constructive solid geometry (CSG, e.g. [Requicha and Voelcker, 1977]) and boundary representation (Brep, e.g., [Baumgart, 1972], [Weiler, 1985]) are two popular representations used by solid modellers to define objects. In CSG, solids are represented by boolean combinations of simple primitive solids. In Brep, solids are represented by a graph embedded in the surface, topologically equivalent to a polyhedron with curved faces. Boundary evaluation is the process which calculates a boundary representation of a solid from it CSG representation.

In this section, we present our geometric representation for three-dimensional solids. This hybrid representation specifies solids as CSG trees of very general primitives (general cylinders, [Binford, 1971]), assembled by using a simple modelling language (section 1.1). Using a general boundary representation, the surfaces of these solids are represented by graphs of trimmed surfaces separated by intersection curves (section 1.2). An efficient and robust boundary evaluator (section 1.3) transforms the CSG represenation into the Brep representation.

### 1-1. CSG representation

In a CSG representation, solids are represented as trees whose leaves are primitives and the other nodes specify set operations between them. We present in this section the CSG representation that we have adopted in our modelling system. We first describe our primitives, which are a large class of generalized cylinders [Binford, 1971], and the way these primitives are represented and built. We then describe our representation for complex objects, and in particular how we specify these objects in terms of a simple modelling language.

#### 1-1-1. Generalized cylinders

A generalized cylinder [Binford, 1971] is the solid obtained by sweeping a planar surface, its cross-section, along a space curve, its axis, or spine. The axis is not necessarily straight, or even planar; the cross-section is not necessarily circular, or even constant; its deformation is governed by a sweeping rule.

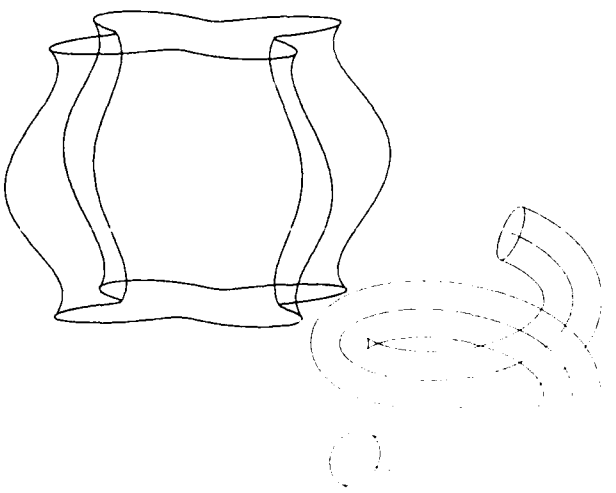


**Figure 3**. Some examples of generalized cylinders implemented in our system.

Generalized cylinders have extensively been used to represent three-dimensional objects in computer vision (e.g., [Marr and Nishihara, 1977], [Brooks, 1981], [Nevatia, 1982], [Binford, Levitt, Mann, 1987], [Ponce, Chelberg, Mann,1987]). Some forms of generalized cylinders (e.g., canal surfaces [Rossignac and Requicha, 1987]) have also been used in solid modelling.

Generalized cylinders are interesting for geometric modelling because they are very general primitives. In particular, they include all the "natural surfaces" (i.e., cuboids, spheres, cylinders, cones, and tori [Rossignac and Requicha, 1987]) usually manipulated by CSG systems, and also, for example, helicoids, and general solids of revolution.

They are also interesting because of their geometric properties: Line drawings of generalized cylinders can be computed analytically (see [Ponce and Chelberg, 1987] and section 3.1). For computer vision applications, it is also possible to prove properties of these line drawings which are independent from the viewing direction [Ponce, Chelberg, Mann, 1987].

Most authors, however, have considered only a very restricted class of generalized cylinders (e.g., the generalized cylinders of *Acronym* [Brooks, 1981] have a straight axis, their cross-section is either a circle or a regular polygon, and the sweeping rule is a linear or bi-linear scaling).

The primitives that we consider here are straight homogeneous generalized cylinders (SHGC's, [Shafer, 1985]), and curved solids of revolution (CSR's). SHGC's are generalized cylinders with an arbitrary cross-section, a straight axis, and an arbitrary scaling sweeping rule. Solids of revolution form a strict subset of SHGC's.

Curved solids of revolution are generalized cylinders with an arbitrary three-dimensional axis, a circular cross-section, and a scaling sweeping rule. Circular tubes, whose circular cross-section is constant, are a strict subset of CSR's. Figure 3 shows some generalized cylinders handled by our modelling system.

Generalized cylinders are volumetric primitives. Both SHGC's and CSR's can be expressed analytically. In a given coordinate frame $(\mathbf{i}, \mathbf{j}, \mathbf{k})$, straight homogeneous generalized cylinders are given by:

$$\mathbf{x}(\mu, s, \theta) = \mu r(s)\rho(\theta)(\cos\theta\mathbf{i} + \sin\theta\mathbf{j}) + s\mathbf{k};$$

$$(\mu, s, \theta) \in [0, 1] \times [s_{min}, s_{max}] \times [0, 2\pi].$$

The polar function $\rho(\theta)$ specifies the shape of the reference cross-section. Similarly, the function $r(s)$ specifies the scaling of this cross-section along the axis. Curved solids of revolution are given by:

$$\mathbf{x}(\mu, s, \theta) = \mathbf{a}(s) + \mu r(s)(\cos\theta\mathbf{n}(s) + \sin\theta\mathbf{b}(s));$$

$$(\mu, s, \theta) \in [0, 1] \times [s_{min}, s_{max}] \times [0, 2\pi],$$

where $\mathbf{n}$ and $\mathbf{b}$ are the normal and the binormal associated to the axis $\mathbf{a}(s)$, and the function $r(s)$ specifies the scaling of the reference cross-section (in this case a unit disc) along the axis.

Notice that these volumetric expressions also give an analytical expression for the surface: in both cases, the "swept surface" (envelope of the swept cross-sections) is given by $\mu = 1$, and the planar "ends" are given by $s = s_{min}$ and $s = s_{max}$.

Notice also that the above expressions of SHGC's restricts slightly the shape of their reference cross-section, which must be star-shaped with respect to the origin. Similarly, the radius of curvature of the reference cross-section of a curved solid of revolution must be smaller than the radius of curvature of its axis (otherwise there may be self-intersections).

For a SHGC, the $\rho$ and $r$ functions, and the corresponding curves $(\rho\cos\theta, \rho\sin\theta)$ and $(s, r(s))$ are in principle arbitrary. However, it is in general convenient to represent them in terms of simple primitive arc segments (in the same way as it is in general convenient to represent complex solids in terms of simple primitives). Similarly, the axis of a CSR can be defined in terms of primitive segments.

In the current implementation, we have two types of primitive arc segments: straight lines and cubic splines. Straight lines are given by their end points. Cubic splines are specified by knot points. These knot points may be of three types: $C^0$ points correspond to orientation discontinuities and are specified by their position plus two tangent directions (one for each side of the point); $C^1$ points correspond to curvature discontinuities and are specified by their position and the tangent direction; $C^2$ points where the curvature is continuous are specified by their position only.

Again, the primitive arc segments can be specified explicitly. However, we find it much easier in most applications to use an interactive graphic editor dedicated to this task. The knot points, their tangents, the types of the primitive segments are all input using a mouse. Simple editing commands (killing a point, moving it, changing its tangents or its type, mirroring a point with respect to some axis, etc..) are available.

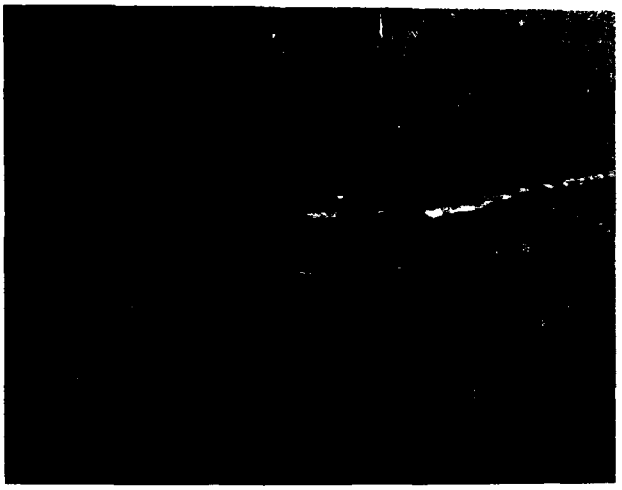


**Figure 4.** A model of the Stanford-JPL hand.

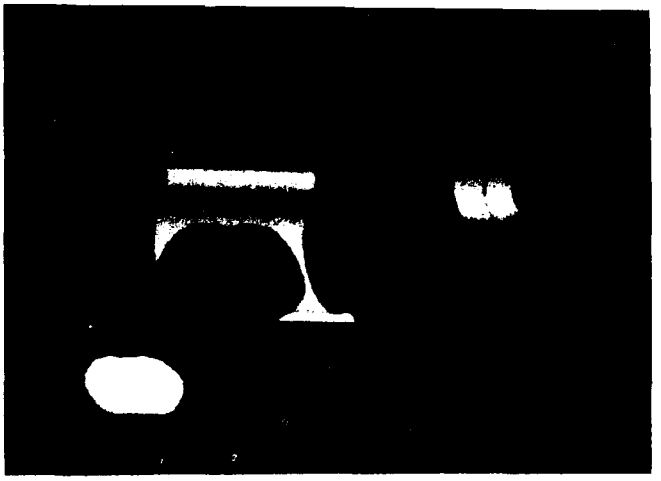


**Figure 5.** A drill. It is specified by one page of code in our modelling language.

At any time, the user can save the current primitive on file, or display it without leaving the editor (using the back to front painting technique of section 3.2). The user can also specify dimensions, e.g., in the case where machine drawings are available. This makes the task of specifying complex primitives very easy.

### 1-1-2. Composite objects

Composite objects are represented by CSG (Constructive Solid Geometry) trees. A CSG tree is a binary tree whose root represent the whole object, and sub-trees its sub-parts. The leaves are the primitives which compose the object. Non-leaf nodes are labelled by the set operations used to combine the primitives.

Primitives may also be associated to non-leaf nodes. They could for example be used to help coarse to fine recognition of an object by representing a "simplified" version of this object [Marr and Nishihara, 1977], or to display an object at different levels of detail depending on its distance from the observer [Crow, 1982].

The relative positions of sub-objects are represented by an affixment graph. The nodes of this graph are primitives, and the arcs represent the geometric transformations between the coordinate systems attached to each primitive. Loops are not allowed in the affixment graph, as they could produce conflicting specifications of the primitives' positions [Ambler and Popplestone, 1975].

The assembly tree and the affixment graph are specified using a simple modelling language. This is straightforward for the assembly tree, but requires more attention for the affixment graph. It is very difficult for a user to model an object by giving the 6 parameters of the transformations between primitives.

Therefore, we have added the possibility of specifying affixments by spatial relationships between planar faces of primitives. The relative positions of the primitives are specified by expressions like "with face < *face1* > in contact with face < *face2* >", which may themselves be parameterized. It is possible to give symbolic values (variables and s-expressions) to the different affixment parameters, so we can model articulated objects (e.g., the three-fingered hand in Figure 4).

Using this simple modelling language, it becomes very easy to specify objects of moderate complexity. In particular, each of the examples presented in this paper (including the drill (Figure 5), hand (Figure 4), and pipe (Figures 1 and 2), which are each made of about a dozen of primitives) can be *specified very concisely* in our modelling language.

## 1-2. Boundary representation

In a boundary representation scheme, a solid is represented by a graph whose nodes are its (curved) faces, and arcs are the (curved) edges separating them (Figure 6). Until now, most boundary representations have been restricted to very simple surfaces and curves.

Recently, *trimmed surfaces* have been proposed to deal with more complex solids (e.g., [Casale, 1987], [Crocker and Reinke, 1987]). They represent faces of solids by surfaces whose original rectangular domain of definition has been trimmed during set operations.

In this section, we present the boundary representation of our modelling system. It uses trimmed surface patches to represent the faces of solids. The trimmed patches themselves are represented by a collection of polygons in parameter space, embedded in a quadtree-like data structure.

### 1-2-1. Trimmed surface patches

A surface is a mapping $\mathbf{x}$ from a closed subset $D$ of $R^2$ into $R^3$. In most solid modellers, the domain of definition $U$ of a surface is a rectangle $I \times J$. When repeated set operations between solids are performed, the domain of definition $D$ of the surface is trimmed, and becomes a strict subset of $I \times J$. The boundaries of $D$ are the back-projections of the intersection curves into parameter space.

The advantage of using a trimmed surface representation is that no approximation is made in representing the surface itself. The only approximations that may occur are in the representation of the 2D boundary of the trimmed domain.

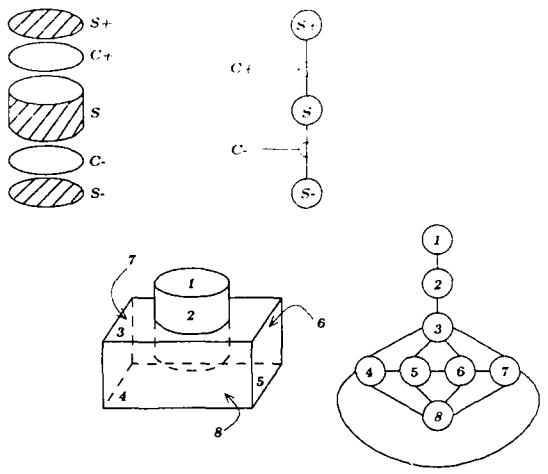



**Figure 6.** The boundary representations for a cylinder and the union of a cube and a cylinder.

Even in that case however, this boundary is guaranteed to lie on the original surface.

Other authors [Casale, 1987] have proposed to represent the trimmed patches by the hierarchy of loops defining the domain $D$. Here, we use an alternative approach. We represent the domain $D$ as a collection of polygons in $I \times J$. These polygons are embedded in a quadtree-like [Samet, 1984] of the rectangle $I \times J$.

The leaves of this quadtree may have three different colors: inside, outside, or intersection. The inside leaves are entirely included within $D$, the outside leaves are entirely included within the complementary of $D$ with respect to $I \times J$. The intersection leaves enclose the boundary of $D$. They store a description of the part of $D$ that they enclose.

Right now, this description is polygonal: boundary curves are represented as polygons, and they split the squares associated to the intersection leaves into a number of polygons that are either completely inside or completely outside the domain $D$. These polygons themselves are marked inside or ourside, so the domain $D$ is formed of the union of all its inside leaves and all the inside polygons of its intersection leaves.

This representation gives us a very simple and efficient description of the trimmed surface patches.

### 1-2-2. Curve segments

A curve is a mapping from an interval $I \subset R$ into $R^3$. A curve segment is defined by a curve and a domain $D \subset I$.

We consider two kinds of curves. The first curves are created during the creation of a solid. They are given by an analytical expression, usually as splines.

The second kind of curves we consider are the intersection curves computed during set operations between solids. They are computed in a polygonal form, but could also be represented as splines whose knot points are the computed intersection points.

## 1-3. Boundary evaluation

Boundary evaluation is the process which calculates the boundary representation of a solid from its CSG representation. A boundary evaluator has normally two main modules. The first one computes the boundary representation of the primitives; the second one computes the result of a given set operation applied to the boundary representation of two solids. The boundary evaluation of a CSG tree is done by recursively calling these two modules.

We now detail these modules. Given the above representation for generalized cylinders, evaluating a primitive is easy. Computing set operations is more difficult. In particular, the two main problems are computational efficiency and robustness in the presence of degenerate cases and round-off errors. We present solutions to these two problems.

### 1-3-1. Evaluating a primitive

The graph corresponding to a given generalized cylinder primitive is built automatically from the specifications of its axis, cross-section, and sweeping rule. For example, the graph associated to a circular cylinder consists of three nodes, its two ends and the swept cylindrical surface; the arcs are the cross-section curves corresponding to $s_{min}$ and $s_{max}$ (Figure 6.a).

Tangent discontinuities in either $\rho$ or $\tau$ partition the surface of a SHGC in a set of smooth faces separated by curves where the normal to the surface is discontinuous. These discontinuities are important for line drawing display, because they are generically observable in images, where they correspond to zero order discontinuities in intensity.

So, if either $\rho$ or $\tau$ have tangent discontinuities, additional nodes must be added to the graph representing a SHGC. Similarly, corresponding new arcs, which are meridians ($\theta$ constant) or parallels ($s$ constant), are added. The case of curved solids of revolution is similar, except that the only discontinuity arcs are parallels.

With our primitive arc segments, tangent discontinuities of $\rho$ and $\tau$ correspond to the end points of line segments and the $C^0$ knot points of cubic arcs. It is therefore trivial to generate automatically the corresponding nodes and arcs. There is evidence that second order surface discontinuities, which correspond to first order discontinuities in intensity, are also detected by humans. It might therefore be interesting to associate to each of the above $C^1$ faces a sub-graph of $C^2$ faces separated by curvature discontinuities. This could be easily done by using the $C^1$ knot points which correspond in fact exactly to these discontinuities.

### 1-3-2. The set operation algorithm

The input of the set operation algorithm consists of a set operation (union, intersection, difference), and two solids $S_1$ and $S_2$. The output of the algorithm is a new solid corresponding to the operation performed.

More precisely, the new solid is obtained by merging the graphs associated to $S_1$ and $S_2$. Some arcs and nodes may be deleted. Conversely, some arcs (new intersection curves) and nodes (a surface patch may be split into smaller disjoint patches) may be created (e.g., Figure 6.b shows the grahs of a cuboid and a cylinder merged to form their union).

In addition, the domain of each parametric patch must be trimmed. To achieve this, our algorithm is divided into three steps: compute the intersection curves in world and

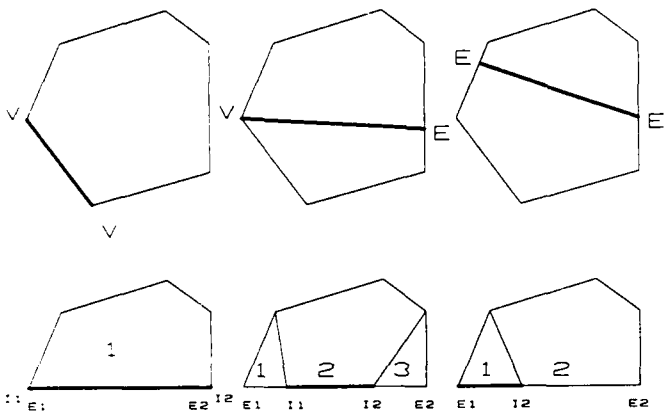



**Figure 7.** The different ways of splitting a convex planar facet.

parameter space; trim the domains of the patches along these curves; update the graph. In each of the steps, our main computational tool will be the quadtree associated to each face of the solids compared.

### 1-3-3. Computing the intersection curves

To find the intersection curves, we use an adaptive subdivision approach (see, e.g., [Carlson, 1982], [Ponce and Faugeras, 1987], for similar approaches). The idea of this method is localize the search for intersections to the regions where they may take place. Given a surface patch, we associate to each node of its quadtree a simple box enclosing the corresponding surface patch.

In the case of patches defined by blending of a set of control points, it is known that a patch is enclosed in the convex hull of the associated control points (see [Miller, 1987]). In a similar way, it is possible to derive analytical expressions for the boxes associated to the class of generalized cylinders considered in our system (this derivation is omitted here for the sake of conciseness, see [Ponce and Healey, 1988], for details).

The algorithm to find the intersection curves is as follows: The roots of the trees associated to the two surfaces are compared first. If they don't intersect then the surface patches don't intersect and the algorithm stops. Otherwise the largest of the two boxes is subdivided, and the recursion proceeds, until the boxes compared are under a pre-determined size.

The result of this step is a list of pairs of intersecting leaves. The associated boxes enclose the intersection curves. To each leaf is associated a list of planar facets which approximate the original surface patch associated to the leaf. For SHGC's and tori, a planar quadrilateral is sufficient. For CSR's whose axis has a non-zero torsion, a pair of triangles is needed.

For each pair of leaves, the corresponding facets are then intersected, and split into non-intersecting facets. One key of the robustness of the set operation algorithm is in this splitting step, described later in detail in a separate subsection. During that step, the edges of each of the non-intersecting facets which correspond to the intersection curves are marked as intersection edges.

This process is repeated for each pair of intersecting leaves. The output of this step is, for each parametric surface, a list of all its intersecting leaves with the associated non-intersecting facets. The intersection curves are made of all the intersection edges of these facets. The next step in the algorithm is to link these edges into curves.

In most solid modelling systems, this linking process is far from trivial as the intersection curves of two algebraic surfaces can be very complex [Farouki, 1986], and heuristics must be used. Here, however, the facets can be easily linked by using a quadtree neighbor-finding algorithm (e.g., [Samet,1982]), and the output of this step is a list of world-space polygons which approximate the intersection curves.

The price to pay for this easy linking is that we have done a polygonal approximation of the intersection curves. Notice however that, within this approximation, the linking is ensured to be correct. The world-space polygons can now be back-projected into parameter space, where they split the domains of definition of the associated parametric surfaces into non-intersecting connected components.

Once again, the back-projected intersection curves are only approximations of the real curves. However, they lie on the real surface, and don't involve any approximation of the surface itself.

### 1-3-4. Trimming the domains

Let us consider a trimmed surface pacth $P$ which is being compared to a solid $S$. Once the intersection curves between $S$ and $P$ have been found, we must use them to trim the domain of definition of $P$.

Let us consider the graph whose nodes are the non-intersecting leaves of the quadtree and the facets associated to the intersecting leaves, and arcs are the neighborood relations between these polygons. The intersection curves delimit a number of connected components of this graph, such that none of these components intersects $S$.

The next step in the algorithm is to classify each of these components as being inside or outside the solid $S$. This could be done by classifying individually each node in the graph, but this is time consuming, as classifying a point is an expensive operation: for example deciding whether a point is inside or outside a general polyhedron has linear complexity in the number of faces.

In fact, classification can be achieved by tracing a ray from the point, intersecting it with the surface, and counting the number of intersections of the ray with the surface (if the number is even, the point is outside, and inside otherwise). This can be done efficiently by using a variant of the ray tracing algorithm described in section 3.3. Moreover, for some primitives (e.g., tori), analytical inside/outside functions are available.

However, the classification of every node would remain expensive, and that is why we use instead the following method. We just classify one point per connected component $C$ and spread its classification to $C$ by using the same

neighbor-finding algorithm as before.

Once all the quadtree nodes and non-intersecting facets have been marked, the nodes and facets that lie outside the result of the set operations can be eliminated. For example, if the union is computed, only all the outside nodes and facets are kept.

### 1-3-5. Updating the graph

Nodes in the surface graph which don't belong to the resulting solid can be immediately destroyed (their quadtree becomes empty). Arcs attached to deleted nodes are deleted too. New nodes are created for each new connected component of the quadtrees.

New arcs are formed by the intersection curves computed during the first step of the algorithm. By maintaining consistent neighborood relations between nodes, facets, and intersection curves during the two first steps, it is then trivial to attach the new arcs to the corresponding nodes.

### 1-3-6. Splitting the facets

One of the big problems in geometric intersection algorithms is to maintain a consistent representation in the presence of degenerate cases (edge-edge intersections, coplanar facets..) and round-off errors. To achieve this goal, we use an approach introduced in [Ladlaw, Trumbore, and Hughes, 1987] for polyhedra.

They present an exhaustive analysis of all types of intersections between convex planar facets, and use it to split two intersecting convex facets into a set of legal non-intersecting facets along the intersection edge. This guarantees that the objects generated by the set operation algorithm are themselves correct.

Here, the planar facets associated to the intersecting leaves of the quadtrees are convex quadrilaterals or triangles, and we can use a similar algorithm. Figure 7 shows the different ways a facet can be split depending where the intersection curves intersect it. The edges corresponding to the intersection curves are the edges used to split the facets, and can therefore be marked during the splitting.

In addition, round-off errors are handled by considering that two points whose distance is less than a given small error are the same. Using both these methods, we have been able to handle correctly cases where one face was intersecting more than twenty other faces, being split in more than one hundred faces in the process.

### 1-4. Complexity issues and results

We claimed earlier that our algorithms were computationally efficient. In this subsection, we briefly justify this claim. We have used the classical analysis of quadtree-based algorithms [Hunter and Steiglitz, 1974] to show that the complexity of steps 1 and 2 is $O(p.2^q)$, where $p$ is the perimeter of the intersection curves in world space, and $q$ is the maximum depth of the quadtree (see [Ponce and Healey, 1987], for details).

This is to be compared to a "naive" alghorithm which would approximate homogeneously the two solids by polyhedra, and whose complexity for a comparable accuracy would be $O(2^{4q}) = O(n^2)$, instead of $O(p\sqrt{n}) = O(p.2^q)$, where $n$ is the number of faces of each polyhedron (for a given solid, $p$ is a constant).

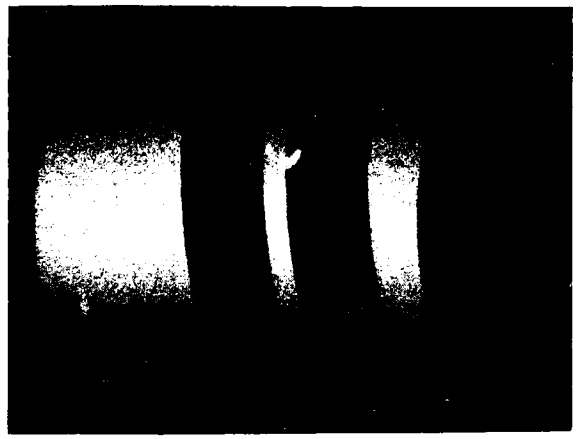


**Figure 8.** A screw, modelled as the difference between a cylinder and a helicoid.

Figures 1,2,4,5, 8 show the results of the boundary evaluation for models made of SHGC's and CSR's.

## 2. Physical Models

The synthesis of realistic images requires models describing the physical world. Appropriate models are readily available. For many years, physicists and optical engineers have studied the optical processes which contribute to image formation. Perhaps the most relevant (and undoubtedly the most complex) of these processes is the interaction of light with matter. Consequently, a large literature exists describing the optical properties of material surfaces. In this section we describe the physical models which our system uses for image synthesis and explain why these models represent a significant improvement over previously used models in computer graphics.

In recent years, computer graphics researchers have used increasingly realistic reflection models. In his pioneering work, Phong [Phong, 1975] introduced a model which represented the light reflected from a surface as a linear combination of a Lambertian diffuse component and a specular component. The intensity of Phong's specular component fell off from its maximum value as the power of a cosine function. Blinn [Blinn, 1977] improved on Phong's model by using a more realistic model for specular reflection based on the work of Torrance and Sparrow [Torrance and Sparrow, 1967] and Trowbridge and Reitz [Trowbridge and Reitz, 1975]. Cook and Torrance [Cook and Torrance, 1981] adopted an even more general model of specular reflection from Beckmann [Beckmann, 1963] and accurately reproduced the spectral properties of the reflected light. In related work, Kajiya [Kajiya, 1985] derived a reflection model for surfaces which are anisotropic scatterers. Recently, Bahar and Chakrabarti [Bahar, 1987] used full-wave theory to describe the light scattered from a rough surface. Their

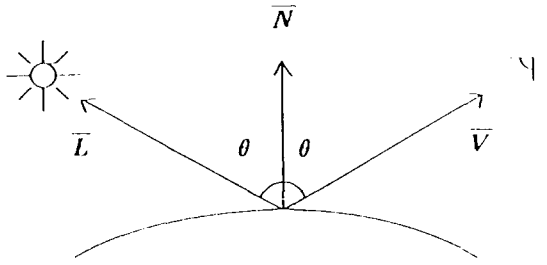



**Figure 9.** Specular Reflection

analysis takes into account the physical optics component of reflection.

An important optical process which has received little attention in computer graphics is the scattering of light from the body of an inhomogeneous material. The scattering of light from the body of a material is also known as colorant layer scattering. It is this colorant layer scattering that largely determines the appearance of the vast majority of the objects we see in everyday life (metal surfaces being the most notable exceptions). For example, plastics, paper, textiles, paints, and ceramics are all inhomogeneous materials for which the dominant optical process is colorant layer scattering.

Most of the work towards improving reflection models has concentrated on interface (specular) reflection and has ignored the scattering of light from inhomogeneous materials. The models of Bahar and Chakrabarti [Bahar, 1987] and Kajiya [Kajiya, 1985] are only relevant for homogeneous materials (e.g. metals and crystals). Blinn's work in [Blinn, 1977] presents an improved model for the specular reflection component. Cook and Torrance [Cook, 1981] take a first step towards modeling body reflection from inhomogeneous materials by treating the scattered light as being colored and uniformly distributed. This simple model, however, does not accurately predict variations in the color and intensity of the reflected light as a function of geometry. In this work, we introduce a model for the scattering of light by colorant layers which substantially improves on existing models. Our model for the scattering of light by inhomogeneous materials is presented in 2.4.

This section is organized into five parts. In 2.1 we describe our representation for the properties of material surfaces. In 2.2 we explain how these physical properties are mapped onto our geometric models. In 2.3, we describe our model for specular reflection which is similar to existing models. In 2.4 we present our model for colorant layer scattering which represents a substantial improvement over existing models. In 2.5 we describe our model for direct and ambient illumination.

### 2-1. Representing Properties of Material Surfaces

Adopting suitable physical models is one of the most important aspects of building a graphics system capable of displaying realistic images. The tendency in computer graphics has been to use physical models that are not sufficiently realistic. There is also the opposite danger of using physical models which are unnecessarily complex. For example, in this work it would be inappropriate and highly inefficient to model light at the level of quantum electrodynamics. In this subsection, we describe our representation for the properties of material surfaces which determine how those surfaces will appear in images.

Our physical models describe intrinsic optical properties of material surfaces. We note that it is typical in computer graphics not to model intrinsic optical properties of materials but rather to model at the level of geometry dependent properties like reflectance. We believe that modeling intrinsic properties is a significant advantage of our approach. Given our representation for the intrinsic properties of material surfaces, there exist reflection models which allow us to determine the image of an arbitrarily shaped specimen of the material under arbitrary viewing and lighting conditions. These reflection models will be discussed in detail in 2.3 and 2.4. In 2.1.1. we distinguish homogeneous and inhomogeneous materials. In 2.1.2. and 2.1.3. we give an overview of reflection and introduce the reflectance model. In 2.1.4. we describe our models for the properties of material surfaces.

#### 2-1-1. Optically Homogeneous and Inhomogeneous Materials

It is useful to divide materials into two classes based on their optical properties. This subsection defines optically homogeneous and optically inhomogeneous materials. The models we use in this work apply to both kinds of materials.

Optically homogeneous materials have a constant index of refraction throughout the material. Consequently, for an object composed of a homogeneous material in air, light undergoes reflection and refraction only as it encounters a surface of the object. Metals and crystals are the most common examples of homogeneous materials.

Optically inhomogeneous materials are composed of a vehicle material with many embedded colorant particles which differ optically from the vehicle. While in the body of an inhomogeneous material, light typically interacts with many colorant particles. The net result of many interactions is that the light is diffused. Some examples of inhomogeneous materials are plastics, paper, textiles, and paints.

#### 2-1-2. Reflection

When light is incident on the surface of an object, some fraction of it is specularly reflected. A smooth surface reflects light only in the direction such that the angle of incidence equals the angle of reflection (Figure 9). The properties of this specularly reflected light are determined by the optical and geometric properties of the surface. For op-

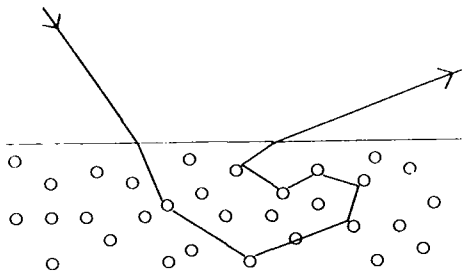


**Figure 10.** Colorant Layer Scattering

tically homogeneous materials which are not transparent, appearance is completely determined by the properties of specularly reflected light. For many inhomogeneous materials, such as plastics, specular effects are also significant.

The fraction of the incident light which is not specularly reflected enters the body of the material. For inhomogeneous materials, the body is composed of a vehicle and many colorant particles. When light encounters a colorant particle, some portion of it is reflected. After many reflections, the light is diffused and a significant fraction can exit back through the surface in a wide range of directions (Figure 10).

### 2-1-3. The Reflectance Model

In subsections 2.3 and 2.4, we develop detailed reflectance models for both specular reflection and colorant layer scattering. In terms of these models, we can quantify the reflectance $R$ of a surface by

$$R(\theta_l, \theta_v, \theta_p, \lambda) = R_S(\theta_l, \theta_v, \theta_p, \lambda) + R_B(\theta_l, \theta_v, \theta_p, \lambda) \quad (2.1)$$

where $R_S$ is the specular reflectance term and $R_B$ is the body reflectance term due to colorant layer scattering. The angles $\theta_l, \theta_v$, and $\theta_p$ are the photometric angles. $\theta_l$ denotes the angle between the surface normal and the illumination direction. $\theta_v$ denotes the angle between the surface normal and the viewer. $\theta_p$ denotes the angle between the illumination direction and the viewing direction. As usual, $\lambda$ denotes wavelength.

The power of the light reflected towards a viewer is given by

$$I(\theta_l, \theta_v, \theta_p, \lambda) = R(\theta_l, \theta_v, \theta_p, \lambda)L(\lambda) \quad (2.2)$$

where $L(\lambda)$ is the spectral power distribution of the light incident on the surface.

### 2-1-4. The Properties of Material Surfaces

We refer to the optically significant part of an opaque object as its material surface. The properties of the light reflected by a material surface are determined by the optical properties of the object's material and the geometric properties of the object's surface. For the purposes of our representation, we keep these material and surface properties distinct. Therefore, a material surface is represented by the combination of a material description and a surface description. In the next two paragraphs, we describe our representations for materials and surfaces respectively.

We represent a material as a record containing five fields. The first field gives the name of the material, e.g. aluminum. The second field specifies whether the material is homogeneous or inhomogeneous and gives a symbolic description of the material's type, e.g. metal, plastic, wood, etc. The third field contains the complex index of refraction of the material as a function of wavelength over the visible range. For an inhomogeneous material, the index of refraction corresponds to the vehicle material. The choice of representation of functions in our system is arbitrary and may be, for example, by means of tables or approximations using basis functions, etc. The fourth field contains the scattering coefficient $\sigma(\lambda)$ of the material as a function of wavelength. The fifth field contains the absorption coefficient $\alpha(\lambda)$ of the material as a function of wavelength. The functions represented in fields four and five apply only to inhomogeneous materials and will be explained in 2.4.

*In our current system,* a surface is described by a record containing the roughness parameter associated with the Torrance-Sparrow specular model [Torrance, 1967]. Our representation allows the possibility of using multiple scale surface roughness as suggested in [Bahar, 1987] and [Cook, 1981].

Our representation for material surfaces has several desirable characteristics. We model only intrinsic, physically meaningful properties of material surfaces. Our representation contains no ad hoc parameters. We do not, for example, split the reflectance into a specular fraction and a diffuse fraction where the two fractions are required to sum to one as is often done. Instead, the amount of specular reflection, body reflection, and absorbed light as functions of wavelength are determined directly from our physical representation for the material surface.

## 2-2. Mapping Physical Properties onto Geometric Models

Given our representation for the physical properties of material surfaces, it is necessary to associate these material surface properties with areas on the surfaces of our geometric primitives. This is done by defining two functions, $f_M$ and $f_S$ over each surface of a generalized cylinder. The function $f_M$ maps a surface point to its material properties record and $f_P$ maps a surface point to its surface properties record. For most applications, we consider $f_M$ and $f_P$ to be piecewise constant functions. In the future, however, we hope to model smoothly changing material and surface properties.

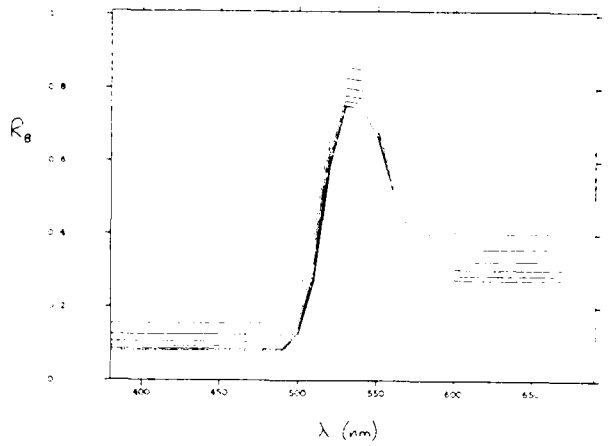



**Figure 11.** The variation of $R_B$ with geometry.

### 2-3. Specular Reflection

When light encounters an interface, some fraction of it is specularly reflected. The Fresnel equations quantify specular reflection from a smooth surface. For rough surfaces, it is possible to augment the Fresnel equations to obtain an appropriate model. This has been done by Torrance and Sparrow [Torrance, 1967]. In 3.3.1. and 3.3.2. we examine these models of specular reflection for monochromatic light. In 3.3.3., we examine how specular reflection affects the spectral properties of incident light. We note that our model for specular reflection is quite similar to the model used by Cook and Torrance [Cook, 1981]. A brief description is included here for completeness.

#### 2-3-1. Fresnel Reflection

At a smooth interface, an incident light ray is reflected in a single direction such that the angle of incidence equals the angle of reflection. Hence, we often measure large values of image irradiance for highlights [Healey, 1987]. A complete theory exists which predicts the properties of specularly reflected light from a smooth air-matter interface in terms of geometry and the fundamental optical properties of the object's surface material. This theory is summarized by the Fresnel equations. We present these equations and discuss some of their implications in this subsection.

The optical properties of a surface material are summarized by the complex index of refraction $M = n - iK_0$ where $n$ is the refractive component and $K_0$ is the absorptive component. Both $n$ and $K_0$ are functions of wavelength. From M, the Fresnel equations completely describe the light reflected from a surface. If unpolarized light is incident at an angle $\theta_i$, then the monochromatic specular reflectance $F$ is

$$F = 0.5(R_\perp + R_\parallel) \tag{2.3}$$

where $R_\perp$ is the perpendicular polarized component and $R_\parallel$ is the parallel polarized component. From electromagnetic theory, $R_\perp$ and $R_\parallel$ are given by

$$R_\perp = \frac{a^2 + b^2 - 2acos\theta_i + cos^2\theta_i}{a^2 + b^2 + 2acos\theta_i + cos^2\theta_i} \tag{2.4}$$

$$R_\parallel = R_\perp \frac{a^2 + b^2 - 2asin\theta_i tan\theta_i + sin^2\theta_i tan^2\theta_i}{a^2 + b^2 + 2asin\theta_i tan\theta_i + sin^2\theta_i tan^2\theta_i} \tag{2.5}$$

where

$$a = 0.5\sqrt{\sqrt{g^2 + 4n^2K_0^2} + g} \tag{2.6}$$

$$b = 0.5\sqrt{\sqrt{g^2 + 4n^2K_0^2} - g} \tag{2.7}$$

$$g = n^2 - K_0^2 - sin^2\theta_i \tag{2.8}$$

Equations (2.4) and (2.5) are known as the Fresnel equations. The Fresnel equations are derived in many places including [Born, 1959].

From the Fresnel equations, it can be shown that for fixed n and fixed $K_0$ the specular reflectance is approximately constant over a large range of incidence angles (roughly $0° \leq \theta_i \leq 70°$) [Sparrow, 1978]. As $\theta_i$ nears $\pi/2$, however, $R_S$ approaches unity for all values of the complex index of refraction. Consequently, as we approach glancing incidence, the color of the specularly reflected light approaches the color of the incident light.

#### 2-3-2. The Torrance-Sparrow Model

The Fresnel equations describe reflection from a smooth surface. In practice, most surfaces are not smooth. The Torrance-Sparrow specular model [Torrance, 1967] describes specular reflection from rough surfaces. This model assumes that a surface is composed of small, randomly oriented, mirror-like facets. Only facets with a normal oriented in the perfect specular direction contribute to the monochromatic specular reflectance $R_S$. The model also quantifies the shadowing and masking of facets by adjacent facets using a geometrical attenuation factor. The resulting specular model is

$$R_S = FDA \tag{2.9}$$

where

$F$ = Fresnel specular reflectance,

$D$ = facet orientation distribution function,

$A$ = geometrical attenuation factor adjusted for foreshortening.

#### 2-3-3. Spectral Properties of Specular Reflection

In general, $R_S$ is a function of $\lambda$. This follows directly from the Fresnel equations and the fact that both components of the complex index of refraction are generally

functions of $\lambda$, i.e. $M(\lambda) = n(\lambda) - iK_0(\lambda)$. Thus for fixed geometry, the function $R_S(\lambda)$ can be computed from the Fresnel equations given $n(\lambda)$ and $K_0(\lambda)$.

The spectral reflectance of an opaque homogeneous material is determined entirely by the specular component of reflectance $R_S$. Therefore a knowledge of $n(\lambda)$ and $K_0(\lambda)$ gives us a complete knowledge of the reflective properties of such a material. For some homogeneous materials, M can vary considerably with wavelength. Copper and gold, for example, reflect long visible wavelengths much more efficiently than short visible wavelengths. On the other hand, the reflectance of aluminum is approximately constant across the visible spectrum.

The body of an inhomogeneous material is made up of colorant particles embedded in a vehicle. While many homogeneous materials, particularly metals, are characterized by a large extinction coefficient $K_0$, inhomogeneous materials have a negligible extinction coefficient across the visible spectrum ($K_0(\lambda) \equiv 0$). For inhomogeneous materials n($\lambda$) depends on wavelength, but this dependence is typically small. For most inhomogeneous materials, n($\lambda$) is constant to less than five percent across the visible spectrum [Kanthack, 1921]. Since both n and $K_0$ are nearly constant for visible light, $R_S$ is constant with respect to $\lambda$. $R_S$ is a function of only geometry for inhomogeneous materials.

### 2-4. Colorant Layer Scattering

For inhomogeneous materials, the most prominent optical process is colorant layer scattering. Light incident on an inhomogeneous material which is not specularly reflected enters the body of the material. In the body of the material, the light is reflected by many colorant particles and becomes thoroughly diffused. Scattering refers to this process of diffusion by many reflections. In this section, we describe a physical model for colorant layer scattering by inhomogeneous materials. Although the theory we describe has recently been used for modeling in a computer vision system [Healey, 1987], we believe that it has not yet been applied to computer graphics.

#### 2-4-1. Kubelka-Munk Theory

Kubelka-Munk (K-M) theory [Kubelka, 1931] is a general mathematical treatment of scattering and absorption in colorant layers. The K-M theory assumes that a colorant layer is composed of a large number of optically identical elementary layers. The thickness of each elementary layer is small compared to the thickness of the entire colorant layer, but is large compared to the diameter of individual colorant particles. Thus it is not necessary to model the optical properties of individual colorant particles. The effects of many colorant particles are modeled by the properties of an elementary layer. An elementary colorant layer is characterized by the parameters $\alpha$ and $\sigma$. $\alpha(\lambda)$ is the fraction of light which is absorbed per unit path length. $\sigma(\lambda)$ is the fraction of light which has its direction changed by scattering per unit path length. Both $\alpha$ and $\sigma$ are functions of wavelength. The model gives rise to simultaneous first order differential equations. These equations can be solved to give expressions for the reflectance and transmission of a colorant layer.

The original Kubelka-Munk theory makes several limiting assumptions. The original theory assumes the boundary condition of diffusely incident light. This is an unrealistic assumption for most real situations. The original K-M theory also assumes that the vehicle containing the colorant particles has an index of refraction equal to that of air. This assumption eliminates the need to consider internal and external reflections at the air-vehicle interface. Unfortunately, this assumption is also not very realistic.

The original Kubelka-Munk theory has been extended by Reichman [Reichman-1, 1973] to eliminate the need for these unrealistic assumptions. Reichman derives an expression for the reflectance of an inhomogeneous material which is valid for collimated light at any angle of incidence. Reichman also uses a method developed by Orchard [Orchard, 1969] to take into account both internal and external reflections at the air-vehicle interface. Experiments have shown that this model accurately predicts the reflecting properties of real materials [Reichman-2, 1973], [Egan, 1979]

The general formulation of Reichman's model is given in [Reichman-1, 1973]. To illustrate the model here, we consider the case of opaque colorant layers composed of isotropic scatterers. For light incident at an angle $\theta_l$, the extended K-M theory describes the body reflectance $R_B$ as

$$R_B = (1 - R'_S)\frac{C(1 - r_i)(R_\infty - D)}{2(1 - r_i R_\infty)cos\theta_l} \qquad (2.10)$$

where $R'_S$ is the fraction of incident light which is specularly reflected. $R'_S$ can be obtained by integrating $R_S$ of (2.9) over the viewing hemisphere. $r_i$ is the internal diffuse surface reflectance approximated by Orchard [Orchard, 1969] as

$$r_i = 1 - \frac{0.5601 - 0.7099n + 0.3319n^2 - 0.0636n^3}{n^2} \qquad (2.11)$$

where n is the index of refraction of the vehicle. Let $w = \frac{\sigma}{(\alpha+\sigma)}$ be the scattering albedo. $R_\infty$ is the reflectance predicted by original K-M for diffusely incident light and is given by

$$R_\infty = \frac{2 - w - 2\sqrt{1 - w}}{w} \qquad (2.12)$$

C and D result from the solution of Reichman's differential equations and are

$$C = \frac{w\cos\theta_l(2\cos\theta_l + 1)}{1 - 4(1 - w)\cos^2\theta_l} \qquad (2.13)$$

$$D = \frac{2\cos\theta_l - 1}{2\cos\theta_l + 1} \qquad (2.14)$$

In contrast to the highly directional properties of specular reflection, colorant layer scattering produces diffusely reflected light. Consequently, $R_B$ in (2.1) is defined relative to

an ideal diffuse surface. One very special case of this model is conservative scattering (also called Lambertian scattering) for which $w(\lambda) = 1$ for all visible wavelengths. A Lambertian scattering model is frequently assumed in computer graphics.

### 2-4-2. Spectral Properties of Colorant Layer Scattering

The color of an inhomogeneous material is primarily due to the scattering and absorbing characteristics of colorant layers. These characteristics are described by the parameters $\alpha(\lambda)$ and $\sigma(\lambda)$. In the limiting case of a Lambertian surface, reflectance is constant with respect to wavelength. On the other hand, the colorant particles in real materials tend to absorb selectively certain wavelengths of light while transmitting others. This selective absorption is the primary cause of the variation of $R_B$ with $\lambda$.

### 2-4-3. An Example

In figure 11 we give an example of the body spectral reflectance predicted by (2.10) for parameters of foliage taken from [Nickerson, 1945]. In the figure, the vertical axis represents $R_B$ in (2.10) with the term $(1 - R'_S)$ excluded so that we can isolate the properties of colorant layer scattering independent of the specular reflection. The various curves in the figure depict the spectral reflectance for several values of the incidence angle $\theta_i$. The values shown are for $\theta_i = 0°, 30°, 45°, 60°, 75°, 90°$. The lowest curve is for $\theta_i = 0°$ and the highest curve is for $\theta_i = 90°$. We see that there is a significant increase in the fraction of light reflected as $\theta_i$ increases. We can also observe a significant change in the color of the reflected light as $\theta_i$ increases. For example, in the blue end of the spectrum, the spectral reflectance varies by a factor of about 2 with geometry, while in the green region, the spectral reflectance varies by a factor of only about 1.1. This variation in both the intensity and the color of the scattered light with geometry is not predicted by other models which are used in computer graphics.

## 2-5. Lighting Models

In this subsection, we discuss our lighting model. In 2.5.1. we describe our representation for sources and in 2.5.2. we describe our approach for dealing with ambient illumination. Our methods include innovations which have not yet appeared in the computer graphics literature.

### 2-5-1. Direct Illumination

We represent a light source as a record containing four fields. The first field gives a name to the source, e.g. point-tungsten-source. The second field specifies the intensity of the source. The third field specifies the source's normalized spectral power distribution. This distribution may be represented as an explicit function (using any function description method) or as a color temperature. The fourth field contains the shape description of the source. For simplicity and computational efficiency, light sources are often represented as points. We do, however, allow arbitrary source shape descriptions using the full power of our geometric models.

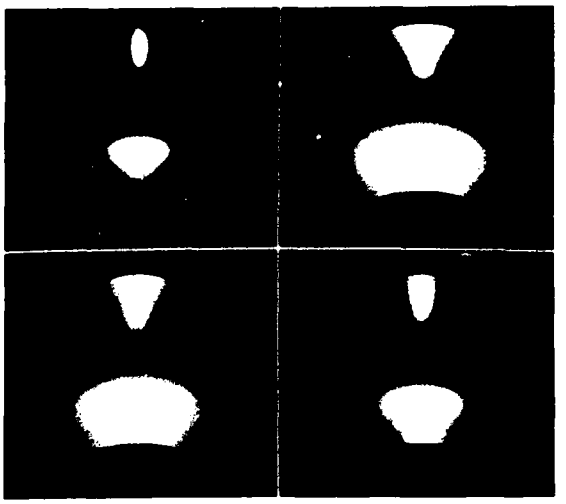


**Figure 12.** See text.

### 2-5-2. Ambient Illumination

In most real scenes there is some amount of ambient illumination. We currently assume ambient illumination that is uniformly distributed. The spectral power distribution of the ambient illumination is taken to be a linear combination of the spectral power distributions of the sources present in the scene.

Despite frequent confusion on this point, the presence or lack of ambient illumination has no effect on the reflecting properties of a surface. To compute the intensity and color of the light reflected from a surface, we consider all light incident on the surface, both direct and ambient. There will be both specular and body reflection of light which is ambiently incident. The specularly reflected ambient light is often neglected by graphics systems, for example see the applications given in [Cook, 1981]. Computing specularly reflected ambient light does, however, improve the quality of images. It causes a color shift in the rendering of glossy inhomogeneous materials that allows for more realistic display. Computing specularly reflected ambient light also allows for more realistic rendering of metals.

## 2-6. Results

We synthesize color images using the algorithm discussed by Wandell in [Wandell, 1987]. The images presented in this paper are displayed on an eight bit color monitor. We expect that our algorithms will be able to produce higher quality images on more sophisticated display devices.

We demonstrate our models in figures 12 and 13. Figure 12 shows four generated images of an object using the spectral properties of copper illuminated by sunlight. Figure 12(a) is copper metal. Figure 12(b) is ceramic. Figure 12(c) is plastic. Figure 12(d) is varnished wood. Figure 13 shows four generated images of an object using the spectral properties of gold illuminated by sunlight. Figure 13(a) is

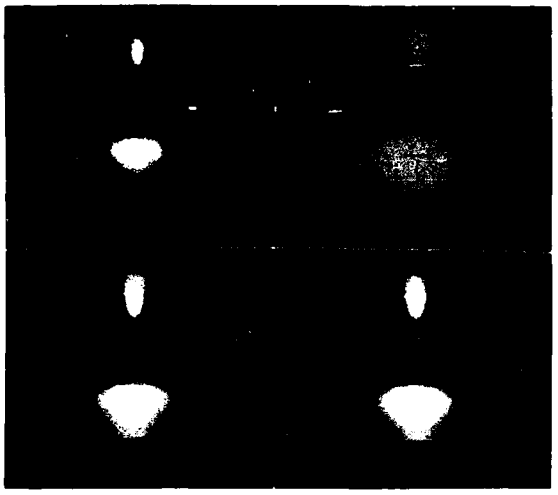


**Figure 13.** See text.

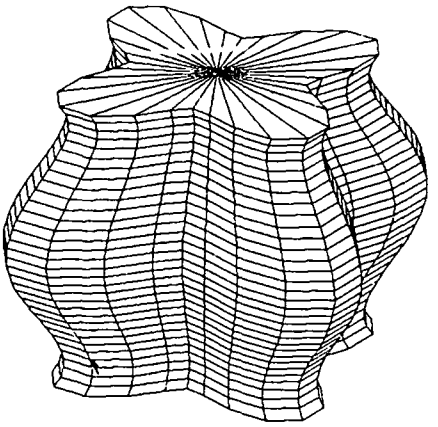


**Figure 14.** A SHGC, rendered by using the back to front painting algorithm.

gold metal. Figure 13(b) is ceramic. Figure 13(c) is plastic. Figure 13(d) is an inhomogeneous material with a larger specular component than in Figure 13(c).

## 3. Rendering

There are multiple ways of rendering solid models. In our modelling system, we use a variety of display algorithms, and most of them are new. The shaded versions of these algorithms use the physical models described in the previous section.

### 3-1. Limbs

Traditionally, the simplest rendering technique is to draw wireframe representations of solids, without any attempt to solve the hidden line/surface problem. In the case of polyhedral objects, this simply means drawing all the polygonal edges.

The case of curved surfaces is more complex. It is not satisfactory to only draw discontinuities, or even a grid of isoparametric lines. In fact, the line drawing of a curved surface should consist of edges (orientation discontinuities) and limbs (occluding contours, where the tangent plane contains the viewing direction).

Edges are simple, they are curves physically drawn on the surfaces. Limbs are more complex, as they depend on the viewing direction. Finding the limbs of an object corresponds to solving the limb equation $\mathbf{v} \cdot \mathbf{n} = 0$, where $\mathbf{v}$ is the viewing direction, and $\mathbf{n}$ is the surface normal.

For simple primitives (e.g., quadrics), analytical expressions for the limbs are readily available. The case of general surfaces is much more difficult. For generalized cylinders, it is often possible, however, to compute analytically the limbs.

Such analytical solutions are available for straight homogeneous generalized cylinders [Ponce and Chelberg, 1987] and curved solids of revolution [Ponce and Healey, 1988]. The primitive examples in Figure 3, as well as the line drawings in the other examples, are rendered by using this method.

### 3-2. Polygonal rendering

Given our representation of solids, it is trivial to build a polyhedral approximation: the facets associated to intersection leaves are already in polygonal form, and the non-intersecting leaves, defined on a square domain, can easily be sampled into quadrilaterals. This approximation can then be rendered by any polygon rendering technique. We have chosen to implement two such algorithms.

The first one is a conventional Z-buffer. Its advantage is speed. However, all the usual problems of z-buffering are present (aliasing, difficulty to compute shadows..). Figures 1,2,4, 5,8, and the figures of the previous section are displayed using this algorithm.

The second algorithm is new. It is based on the same idea as Fuchs' back to front painting algorithm: the idea is to sort the facets so that the first facet displayed is the farthest from the observer, and the last one is the closest. This way, there is no need for z-buffering, and the rendering can be very efficient on computers that have fast polygon painters.

To sort the faces, we use again the associated quadtree. The two subdividing planes of a given node are used to sort the four sons of this node. All the facets can be displayed from back to front by visiting the tree in a depth first manner according to this sort.

The algorithm is quite efficient, and allows shaded rendering of isolated primitives in near real time on a lisp machine. The present algorithm only works for isolated primitives (the subdividing planes don't separate different primitives). It should be possible to extend the algorithm to non-intersecting primitives which can be separated by a plane.

Figure 14 shows an example of SHGC rendered using this algorithm on a black and white lisp machine. This algorithm is especially useful during the interactive design

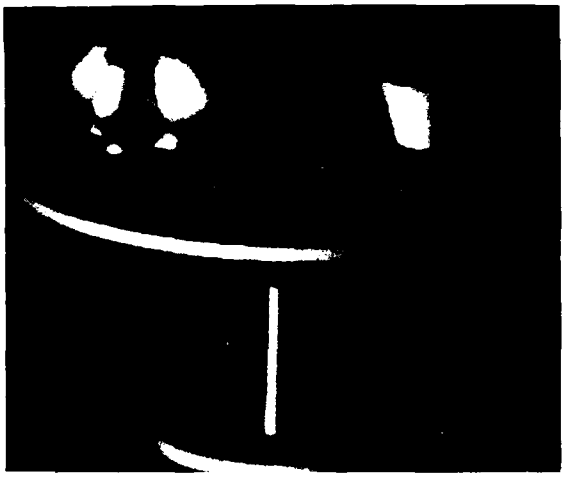


**Figure 15.** A simple scene rendered by our ray tracing algorithm for shaded images.

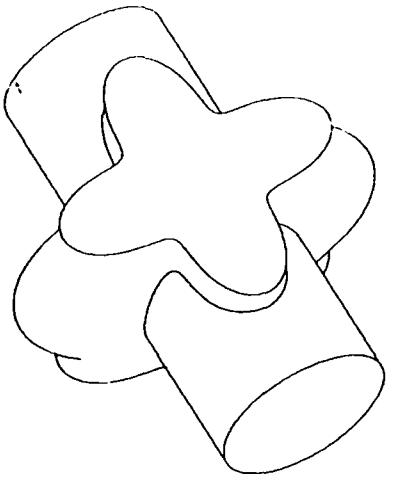


**Figure 16.** The union of two generalized cylinders. It is rendered by using ray tracing at limbs, edges, and intersection curves.

of new primitives (see section 1.1.1).

### 3-3. Ray tracing

Shaded ray tracing remains the most flexible and realistic rendering technique. Here, we use again the quadtree representation of the surface patches to get acceptable computing times. The algorithm is analogous to Kajiya's algorithm (see [Kajiya, 1983], and also [Ponce and Faugeras, 1987]).

Individual rays are intersected with the tree associated to a given surface patch. If the ray intersects the corresponding box, the box is subdivided, and the recursion proceeds. Otherwise the subdivision stops, there is no intersection.

Usually, the boxes associated to the sons of a node may intersect, and cannot therefore be sorted along the ray. In that case, it is possible that several branches of the tree leading to faces obscuring each other may have to be visited.

In certain cases, however (e.g., in the case of SHGC's), the boxes don't intersect, and it is possible to sort them along the ray from front to back. This ensures that only one branch of the tree is visited for each surface patch.

The complexity of the algorithm can be shown to be $O(N.q)$, where $N$ is the number of pixels and $q$ is the maximum depth of the tree (see [Ponce and Healey, 1988] for details). This should be compared to the $O(N.2^{2q})$ that would be achieved by using conventional methods.

For additional speed-up, the screen itself is organized in a quadtree. This means that the incident rays are intersected with only a few patches. During shadow computation, however, such techniques cannot be used, and this is when our algorithm proves to be specially useful.

Figure 15 shows a ray-traced scene made of a few generalized cylinders. The scene includes shadows. It has been computed at a $512 \times 512$ resolution, antialiasing being taken care of by using double resolution at the boundaries between surface patches. The computing time for this scene (equivalent to 40,000 polygons) is of three hours on a lisp machine.

### 3-4. Ray tracing at limbs

A variant of this method is to use ray tracing for line drawing display. The idea is to compute the limbs and edges as for wireframe display, and only do the ray tracing at the contour pixels.

This method reduces the complexity of ray tracing by a factor of $S/P$ where $S$ is the total projected area of the objects drawn, and $P$ is the total projected perimeter of their contours. This means that its complexity is roughly $O(\sqrt{N}.q)$ (area grows as the square of perimeter when resolution increases).

This technique as been used in the line drawing of Figure 16, which shows the union of two generalized cylinders. The computing time here is of less than 5mn on a lisp machine.

A last line drawing method is even more efficient, as it reduces the amount of ray tracing computations to one per parametric patch. The idea of the algorithm is that the visibility of image contours changes only at T-junctions, where two contours cross each other, and cusps, where the contour obscures itself.

T-junctions can be found during the actual drawing of the lines, at screen resolution. As limbs, cusps can be computed analytically for generalized cylinders (see [Ponce and Chelberg, 1987]). At T-junctions and cusps, it is necessary to evaluate the depth and decide which contour segment is visible (analogous to a pointwise z-buffer). The visibility of this contour will not change until the next junction.

It is in fact necessary to add one ray tracing test per object, as it is possible that a patch is completely obscured by an other one, without any junction. This method has been implemented in an earlier version of the modelling system, using simpler primitives, and allowed us to reach near real-time performances for scenes composed of a dozen primitives.

Figure 17 shows an example of scene rendered using this technique. We are currently working on re-implementing

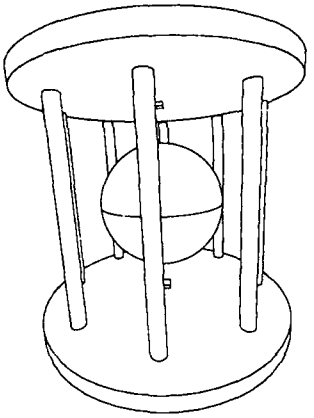


Figure 17. A scene rendered by finding T junctions and cusps.

this algorithm in the current version of the modelling system.

## Acknowledgments

We would like to thank David Shen and Bruno Olshausen for substantial programming help. We would also like to thank the members of the group who helped with the development of the geometric modelling system including Tom Binford, David Chelberg, David Kriegman, and Wallace Mann.

## References

1. Ambler, A.P., Popplestone, R.J., "Inferring the positions of bodies from specified spatial relationships," Artificial Intelligence 6 (1975), pp. 157-174.
2. Bahar, E. and Chakrabarti, S., "Full-Wave Theory Applied to Computer-Aided Graphics for 3D Objects." *IEEE Computer Graphics and Applications* (July 1987), pp. 46-60.
3. Baumgart, B., "Winged-edge polyhedron representation", Stanford AI Report, No. CS-320 (1972).
4. Beckmann, P. and Spizzichino, A., *The Scattering of Electromagnetic Waves from Rough Surfaces*, MacMillan, (1963).
5. Binford, T.O., Levitt, T., Mann, W., "Bayesian inference in model-based machine vision," Proc. Workshop on Uncertainty in Artificial Intelligence, (1987).
6. Blinn, J., "Models of Light Reflection for Computer Synthesized Pictures," *Computer Graphics*, Vol. 11, No. 2, Proceedings of SIGGRAPH-77 (July 1977), pp. 192-198.
7. Born, M. and Wolf, E., *Principles of Optics*, Pergamon Press, New York, (1959).
8. Brooks, R.A., "Symbolic reasoning among 3D models and 2D images," Artificial Intelligence 17 (1981), pp. 285-348.
9. Casale, M.S., "Free-form solid modeling with trimmed surface patches," IEEE Comp. Graphics and Application, Vol. 7 No. 1, (1987), pp. 33-43.
10. Carlson, W.E., "An algorithm and data structure for 3D object synthesis using surface patch intersections," Proc. SIGGRAPH 82 (1982), pp.255-264.
11. Cook, R. and Torrance, K., "A Reflectance Model for Computer Graphics," *Computer Graphics*, Vol. 15, No. 3, Proceedings of SIGGRAPH-81 (August 1981), pp. 307-316.
12. Crocker, G.A., Reinke, W.F., "Boundary evaluation of non-convex primitives to produce parametric trimmed surfaces," Proc. SIGGRAPH 87 (1987), pp. 129-136.
13. Crow, F.C., "A more flexible image generation environment," Proc. SIGGRAPH 82 (1982), pp. 9-18
14. Egan, W. and Hilgeman, T., *Optical Properties of Inhomogeneous Materials*, Academic Press, New York, (1979).
15. Farouki, R.T., "The characterization of parametric surface sections," Comp. Vis. Gr. Im. Proc., Vol. 33 (1986), pp. 209-236.
16. Healey, G. and Binford, T.O., "The Role and Use of Color in a General Vision System," Proceedings of ARPA Image Understanding Workshop, USC (1987).
17. Hunter, G.M, and Steiglitz, K., "Operations on images using Quadtrees", IEEE Trans. Patt. An. Machine Int., Vol. PAMI-1, No. 2 (1979).
18. Kajiya, J., "Anisotropic Reflection Models," *Computer Graphics*, Vol. 19, No. 3, Proceedings of SIGGRAPH-85 (July 1985), pp. 15-21.
19. Kanthack, R., *Tables of Refractive Indicies*, Vol. II, App. III, Hilger, London (1921).
20. Kubelka, P. and Munk, F., "Ein Beitrag sur Optik der Farbanstriche," Z. tech. Physik., 12, 593 (1931).
21. Ladlaw, D.H., Trumbore, W.B., Hughes, J.F., "Constructive solid geometry for polyhedral objects," Proc. SIGGRAPH-86 (1986), pp. 161-170.
22. Marr,D., and Nishihara,K., "Representation and recognition of the spatial organization of three dimensional shapes," Proc. Royal Soc. of London, B-200 (1977), pp. 269-294.
23. Miller, S., "Sculptured surfaces in solid models: issues and alternative approaches," IEEE Comp. Graphics and Application, Vol. 6 No. 12, (1986), pp. 37-47
24. Nickerson, D., Kelley, K., and Stultz, K., "Color of Soils," *Journal of the Optical Society of America*, 35 (1945), pp. 297-300.

25. Orchard, S. "Reflection and Transmission of Light by Diffusing Suspensions," *Journal of the Optical Society of America*, Vol. 59, No. 12 (December 1969), pp. 1584-1597.

26. Phong, B. "Illumination for Computer Generated Pictures," *Communications of ACM*, Vol. 18, No. 6 (June 1975), pp. 311-317.

27. Ponce, J., and Healey, G., "Using generic geometric and physical models for image synthesis", Stanford Univ. Robotics Lab. Tech. Rep. (1988) to appear.

28. Ponce, J., and Chelberg, D., "Finding the limbs and cusps of generalized cylinders," International Journal of Computer Vision, Vol. 1 No. 3 (1987).

29. Ponce, J., Chelberg, D., and Mann, W., "Invariant properties of the projections of straight homogeneous generalized cylinders," Proc. of the First International Conference on Computer Vision, London (1987).

30. Reichman, J., "Determination of Absorption and Scattering Coefficients for Nonhomogeneous Media. 1: Theory," *Applied Optics*, Vol. 12, No. 8 (August 1973), pp. 1811-1815.

31. Reichman, J. "Determination of Absorption and Scattering Coefficients for Nonhomogeneous Media. 2: Experiment," *Applied Optics*, Vol. 12, No. 8 (August 1973), pp. 1816-1823.

32. Requicha, A., and Voelcker, H., "Constructive solid geometry", Tech. Memo 26, Production Automation Project, Univ. of Rochester, N.Y., (1977).

33. Rossignac, J.R., and Requicha, A.G., "Piecewise-circular curves for geometric modeling," IBM J. Res. Develop., Vol. 31, No. 3 (1987).

34. Samet, H., "The Quadtree and related hierarchical data structures," Computing Surveys, Vol. 16, No 2 (1984), pp. 187-260.

35. Samet, H., "Neighbor finding techniques for images represented by Quadtrees", Comp. Gr. Im. Proc., Vol 18 (1982).

36. Shafer, S.A., **Shadows and silhouettes in computer vision**, Kluwer Academic Publishers (1985).

37. Sparrow, E. and Cess, R., *Radiation Heat Transfer*, McGraw-Hill, New York (1978).

38. Torrance, K. and Sparrow, E., "Theory for Off-Specular Reflection from Roughened Surfaces," *Journal of the Optical Society of America*, 57 (1967), pp. 1105-1114.

39. Trowbridge, T. and Reitz, K., "Average Irregularity Representation of a Roughened Surface for Ray Reflection," *Journal of the Optical Society of America*, 65 (May 1975), pp. 531-536.

40. Wandell, B., "The Synthesis and Analysis of Color Images," IEEE Transactions on Pattern Analysis and Machine Intelligence, 9, 1 (Jan. 1987), pp. 2-13.

41. Weiler, K., "Edge-based data structures for solid modelling in curved-surface modeling environments", IEEE CG and A, Vol. 5, No. 1 (1985).